TRIUMPH ADLER

TA Computer Systeme

TA 1600/30 T A X O

System Command Interpreter

01.06.1983

TA 1600

TA 1600/30 T A X O

System Command Interpreter

01.06.1983

ab TAXO V200

INHALTSVERZEICHNIS

1 DER SYSTEM-COMMAND-INTERPRETER UND SEINE ANWENDUNG

1.1 Übersicht

2 BEISPIELE UND BESONDERE AUSDRRÜCKE

2.1 Beispiel einer Prozedur

2.2 Erstellen einer Prozedur

2.3 Besondere Ausdrücke
2.3.1 Eingabeanforderung (Prompt)
2.3.2 Der Kommando-Modus
2.3.3 Kommandoprozedur
2.3.4 Anfangswerte
2.3.5 Sonderzeichen
2.3.6 Schlüsselwort (Keyword)
2.3.7 Prozedurnamen
2.3.8 Kommando-Prozessor
2.3.9 Primitive-Kommandos
2.3.10 Vordergrund-Modus (Foreground-Mode)
2.3.11 Hintergrund-Modus (Background-Mode)
2.3.12 .BID, .QBID, .DBID
2.3.13 Stapelverarbeitung (Batch Stream)
2.3.14 TLF (Terminal Local File)
2.3.15 Terminal Communication Area
2.3.16 Privilege Levels
2.3.17 Menüs

3 STANDARD-DATEN

3.1 DIE DATEI .S$NEWS
3.2 .S$PROC.M$00 und .S$PROC.M$01
3.3 Menüdaten - .S$PROC.M$LC

4 DIE SCI SPRACHE

4.1 Prompts (Eingabeanforderungen)
4.1.1 Promptttyp ACNM
4.1.2 Promptttyp INT
4.1.3 Promptttyp NAME
4.1.4 Promptttyp STRING
4.1.5 Promptttyp YES
4.1.6 Abkürzungen für Prompts
4.1.7 Synonyme
4.1.8 Auswertungen von Synonymen
4.1.9 Benutzung des Dollar-Zeichens in Benutzereingaben

INHALTSVERZEICHNIS (Fortsetzung)

4.2 SCI Primitiv Kommandos
4.2.1 Primitiv Kommandos .PROC und .EOP
4.2.2 Primitiv Kommandos .BID, QBID und .DBID
4.2.3 Primitiv Kommandos .DATA und .EOD
4.2.4 Primitiv Kommando .EVAL
4.2.5 Primitiv Kommando .EXIT
4.2.6 Primitiv Kommando .IF, .ELSE und .ENDIF
4.2.7 Primitiv Kommandos .LOOP, .UNTIL, .WHILE und .REPEAT
4.2.8 Primitiv Kommando .MENU
4.2.9 Primitiv Kommando .OPTION
4.2.10 Primitiv Kommando .OVLY
4.2.11 Primitiv Kommando .PROMPT
4.2.12 Primitiv Kommando .SHOW
4.2.13 Primitiv Kommando .SPLIT
4.2.14 Primitiv Kommando .STOP
4.2.15 SVC Primitiv
4.2.16 Primitiv Kommando .SYN
4.2.17 Primitiv Kommando .USE

5 AUSFÜHRUNG DES SCI

5.1 Namensbildung für Dateien und Directories

5.2 Stapelverarbeitung (Batch Streams)
5.2.1 Beispiel für eine Batch-Kommandofolge
5.2.3 Die Ausführung des Batch
5.2.4 Die Batch-Listing
5.3 Expert Mode

1 DER SYSTEM-COMMAND-INTERPRETER UND SEINE ANWENDUNG

1.1 Übersicht

SCI ist die Abkürzung für "System-Command-Interpreter". Er ist die Schnittstelle zwischen dem TAXO-Benutzer und dem TAXO Betriebssystem. Der SCI verarbeitet die Anforderungen des Benutzers für TAXO Betriebsmittel und Routinen, nimmt die für die Steuerung von Benutzer- und Steuerprogrammen notwendigen Parameter auf und stellt dem Benutzer die für die Erstellung von Steuerroutinen und -prozeduren notwendigen Mittel zur Verfügung, die es ihm ermöglichen, auf diese durch die Eingabe einfacher Kommandos über Terminal-Tastatur zuzugreifen.

Wie aus seinem Namen hervorgeht ist der SCI ein Interpreter der Systemkommandos. SCI-Prozeduren und SCI-Kommandos werden mit den gleichen Techniken programmiert, die auch für COBOL, BASIC oder andere Programmiersprachen angewendet werden. Die Sprache des SCI setzt sich aus den gleichen Elementen zusammen wie andere Programmiersprachen auch: Anweisungen für die Definition der Daten, logische Kommandos, Eingabe/Ausgabekommandos und arithmetische Kommandos. Die verschiedenen Elemente dieser Sprache werden dazu benutzt, Kommandoprozeduren zu erstellen und diese auf einer Platte zu speichern. Nach Eingabe des Namens einer bestimmten SCI-Prozedur als Antwort auf eine SCI-Meldung sucht der Interpreter die entsprechende Datei aus der Prozedurbibliothek und führt die in der Prozedurdefinition angegebenen Anweisung aus, um die programmierte Funktion durchzuführen. Die TAXO SCI-Standardkommandos, z.B. SF (Show File) oder DF (Delete File) sind in der SCI-Sprache programmiert und werden als Prozeduren in der Prozedurbibliothek .S$PROC abgespeichert.

Nach Durchsicht folgender Beispiele können erfahrene Benutzer direkt zum Kapitel "die SCI-Sprache" umblättern.

2 BEISPIELE UND BESONDERE AUSDRÜCKE

Die folgenden Beispiele und Definitionen dienen dem Verständnis der SCI-Sprache und ihrer Implementation.

2.1 Beispiel einer Prozedur

SF (Show File) ist eines der dem Benutzer verfügbaren SCI Standardkommandos. Das Kommando zeigt den Inhalt einer Plattendatei am Bildschirm des interaktiven Terminals an. Das folgende Beispiel zeigt dem Benutzer eine Möglichkeit, das SF-Kommando mittels der SCI-Sprache zu programmieren:

Zeilen Nr.	Daten von der Plattendatei
1	.PROC SF (SHOW FILE),
2	FILE PATHNAME = ACNM@ SFP)
3	.SHOW &FILE PATHNAME
4	.EOP

Die im Beispiel angegebenen Zeilennummern sind nicht ein Teil der Sätze der Datei. Sie dienen nur als Bezugspunkt für die folgende Erläuterung.

Eine ähnliche Datei ist auch Teil der .S$PROC Bibliothek. Nach Eingabe des SF-Kommandos am Terminal und anschließendem Drücken der RETURN-Taste sucht der SCI die Datei namens SF in der Prozedurbilbiothek, greift auf diese Datei zu und interpretiert die in der Prozedur vorhandenen Kommandos, um die notwendige Funktion auszuführen.

MERKE

Der SCI verarbeitet die Anweisungen die fast wie normale englischsprachige Sätze anmuten in der Form, in der sie im Beispiel erscheinen. SCI-Prozeduren werden nicht in eine Maschinensprache umgewandelt.

Sinngemäß führt der SCI folgende Schritte aus:

1. Aufnahme des vom Benutzer angegebenen Prozedurnamens (SF). Ein Prozedurname kann aus bis zu 8 Zeichen bestehen.

2. Der SCI bildet den Pfadnamen für die Prozedur aus dem Prozedurnamen und dem Namen der gegenwärtigen Prozedurbibliothek. In unserem Beispiel hat die Prozedurbibliothek den Namen .S$PROC; der Pfadname für die Datei ist daher .S$PROC.SF.

3. Der SCI sucht die Prozedurdatei in der Prozedurbibliothek.

4. Er liest die Daten aus der Prozedurbibliothek.

5. Er führt eine Überprüfung durch, um festzustellen, daß er die richtige Prozedur gefunden hat, indem er das eingegebene Kommando mit dem Prozedurnamen SF, der dem vollen Namen auf Zeile 1 der Prozedur vorausgeht, vergleicht. Zeile 1 ist äquivalent zu der Definition eines Programmnamens ohne PROC.

6. Der SCI zeigt die in der Prozedur definierten Benutzermeldung(en) am Bildschirm an. In diesem Fall wird vorher der Inhalt des Bildschirms gelöscht und der volle Name der Routine, d.h. SHOW FILE, in der linken oberen Ecke des Bildschirms angezeigt. Der Klammerausdruck hinter SF auf der Zeile 1 ist der volle Name der Prozedur. Wenn vorhanden, wird der volle Name der Prozedur am Bildschirm angezeigt.

   In unserem Beispiel zeigt der SCI nach Löschen des Bildschirms folgenden Text:

```
SHOW FILE
        FILE PATHNAME:
```

   In diesem Beispiel ist der Text FILE PATHNAME: eine Eingabeanforderung. Eingabeanforderungen werden als Teil der Zeile definiert, in der der Prozedurname angegeben wird (Zeile 1). Der SCI benutzt das Komma, um auf eine Fortsetzung der Zeile hinzudeuten. Zeile 2 ist daher eine Fortsetzung der Zeile 1, da das letzte Zeichen ungleich Blank in der Zeile 1 das Kommazeichen ist. Würde eine Anzahl Parameter benötigt werden, so würde das Kommazeichen auch die Zeile 2, sowie eine oder mehrere der anschließenden Zeilen abschliessen. Die Zeile, die die letzte Eingabeanforderung enthält, wird ohne Komma abgeschlossen.

   Das Beispiel für SF könnte auch wie folgt geschrieben werden:

```
SF (SHOW FILE), FILE PATHNAME = ACNM (@$SF$P)
      .SHOW & FILE PATHNAME
```

   Für bessere Lesbarkeit wird normalerweise jede Eingabeanforderung auf einer separaten Zeile geschrieben.

7. Nun kann der Benutzer die notwendigen Parameter über Tastatur eingeben. Der Antwortmodifikator ACNM (Access Name), der dem Gleichheitszeichen folgt informiert den SCI, daß die auf die Eingabeanforderung FILE PATHNAME zu erwartende Antwort ein Dateiname sein muß. Auf diese Weise wird es dem SCI ermöglicht, die syntaktische Richtigkeit der Eingabe zu überprüfen. Tabelle 4-1 enthält eine Liste aller Antwortmodifikatoren und ihre Funktionen.

8. Der SCI führt die anderen in der Prozedur spezifizierbaren Operationen aus. In diesem Fall wird die .SHOW Primitive (eine Primitive ist ein Basiskommando in der SCI-Sprache) aufgerufen um die Daten der als FILE PATHNAME eingegebenen Datei anzuzeigen. Das dem FILE PATHNAME vorausgehende und-Zeichen (&) besagt, daß dieser eine Eingabeanforderung ist und daß der Text FILE PATHNAME durch Eingabe des Pfadnamens der entsprechenden Datei zu ersetzen ist.

MERKE

Der Ausdruck @SFP, der auf Zeile 2 der Prozedur dem Text ACNM folgt bewirkt, daß der Wert des FILE PATHNAME ursprünglich auf den Wert des Synonyms @SFP gesetzt wird.

Zusammenfassung:

Zeile 1 der Prozedur definiert den Namen der Prozedur und darf eine Zeichenkette enthalten, die vom SCI am Bildschirm angezeigt wird, wenn die Prozedur aufgerufen wird.

Zeile 2 ist eine Datendefinition für die Eingabeanforderung FILE PATHNAME, die als Dateiname definiert ist. Die Eingabeanforderung FILE PATHNAME hat den Anfangswert des Synonyms @SFP.

Zeile 3 ist eine Anweisung, die eine Bildschirmanzeige der in der hinter FILE PATHNAME angegebenen Datei gespeicherten Daten auslöst.

2.2 Erstellen einer Prozedur

Prozedur SF ist in der Systemprozedur-Bibliothek als sequentielle Textdatei gespeichert. Der Text-Editor kann für das Erstellen und Modifizieren von SCI-Prozeduren in der gleichen Weise eingesetzt werden, wie für das Erstellen und Editieren von Quellcodeprogrammen in COBOL, FORTRAN oder anderen Programmiersprachen.

MERKE

Das Editieren einer SCI-Prozedur, um so eine neue Prozedur zuwege zu bringen, wird nicht empfohlen.

Das Beispiel SHOW FILE kann wie folgt in das System übernommen werden:

1. Aufrufen des Text-Editors durch XE (Execute the Editor) odes XES (Execute the Editor with Scaling). Keine "File Access Name" eingeben.

2. Die folgenden vier Zeilen eingeben. Jede Zeile mit RETURN abschliessen.

```
.PROC SF(SHOW FILE)
        FILE PATHNAME = ACNM(@$SF$P)
        .SHOW &FILE PATHNAME
.EOP
```

MERKE

Das Einrücken von Zeilen verbessert die Lesbarkeit einer Prozedur für den Zweck der Protokollierung. Der SCI ignoriert solche Leerstellen.

3. CMD-Taste drücken.

4. QE eingeben und RETURN drücken.

5. RETURN drücken als Antwort auf die Eingabeanforderung ABORT?:

6. .S$PROC.EXP eingeben und RETURN drücken als Antwort auf die Eingabeanforderung OUTPUT FILE:

7. Y eingeben und RETURN drücken als Antwort auf die Eingabeanforderung REPLACE?:

8. RETURN drücken als Antwort auf die Eingabeanforderung MOD LIST:

   Die oben genannten Schritte erstellen eine Datei namens .S$PROC.EXP mit den 4 Beispielzeilen für SF in der Systemprozedur-Bibliothek .S$PROC.

9. Nach Wiedererscheinen der SCI-Eingabeanforderung, EXP eingeben und RETURN drücken. Dadurch wird der SCI aufgefordert die Kommandoprozedur EXP zu suchen und sie auszuführen. Der folgende Fehler wird unmittelbar gemeldet:

   FF02 PROC LIBRARY ERROR

   Der Fehler wurde durch die Abweichung zwischen dem in Zeile 1 der Prozedur angegebenen Prozedurnamen und dem Namen der Prozedurdatei verursacht. Zeile 1 definiert SF als Prozedurnamen, aber der Pfadname ist .S$PROC.EXP

MERKE

Häufig auftretende SCI Prozedurfehler sind am Ende dieses Kapitels aufgelistet. Für eine vollständige Liste der Fehlermeldungen siehe Tabelle 3-3 im "TAXO-Fehlerhandbuch".

10. Um diesen Fehler zu beheben, CMD drücken, um einen Rücksprung zur SCI Eingabeanforderung zu bewirken; anschließend XE eingeben und RETURN drücken. Der Dateiname erschient automatisch hinter der Eingabeanforderung FILE ACCESS NAME:. RETURN drücken.

11. Den Namen der Prozedur in Zeile 1 in EXP abändern; anschließend die CMD Taste und das QE Kommando benutzen, um eine neue Ausgabedatei zu erstellen. Für eine Beschreibung der Arbeitsweise des Text-Editors siehe "TAXO - Text Editor Handbuch".

    Die neue Prozedur sollte folgendes Aussehen haben:

```
.PROC EXP(SHOW FILE),
          FILE PATHNAME = ACNM (@$SF$P)
          .SHOW &FILE PATHNAME
.EOP
```

12. EXP eingeben und RETURN drücken als Antwort auf die SCI Eingabeanforderung. Der Name der Prozedur, d.h. SHOW FILE und die Eingabeanforderung FILE PATHNAME sollten am Bildschirm erscheinen.

13. .S$PROC.EXP eingeben und RETURN drücken. Der Inhalt der Prozedurdatei wird am Bildschirm angezeigt. CMD drücken um zur SCI Eingabeanforderung zurückzuspringen.

2.3 Besondere Ausdrücke

Besondere, nur in Verbindung mit TAXO benutzte Ausdrücke werden in den Beschreibungen der SCI Funktionen und Kommandos immer wieder gebraucht. Manche dieser Ausdrücke haben jedoch mehr als eine Bedeutung. Diese Ausdrücke sind in den folgenden Abschnitten erklärt.

2.3.1 Eingabeanforderung (Prompt)

In dieser Beschreibung wird der Ausdruck "Eingabeanforderung", oder alternativ das Wort "Prompt" für den SCI-Prompt () sowie für die von den Kommandos angeforderte Parametereingabe benutzt. Der SCI zeigt diese Prompts am Bildschirm an wenn der Benutzer ein Kommando aufruft. Die Prompts deuten auf die Parameter hin, die für das Ausführen der durch das SCI Kommando angeforderten Operation benötigt werden.

Beispiel eines Prompts:

()SF	SCI Prompt und Benutzerantwort
SHOW FILE	Der SCI zeigt den vollen Namen an
FILE PATHNAME	Der SCI fordert die Eingabe des anzuzeigenden Dateinamens an

2.3.2 Der Kommando-Modus

Der SCI befindet sich im Kommando-Modus, wenn der SCI-Prompt () am Bildschirm eines Terminals angezeigt wird. In diesem Modus faßt der SCI alle Benutzereingaben als Kommandos auf, d.h. nicht als Daten.

MERKF

Das SCI Kommando-Prompt () kann vom Benutzer geändert werden. Dieser Vorgang wird in einem späteren Kapitel beschrieben. Beispiele anderer SCI Kommando-Prompts sind auf der TAXO-Systemplatte implementiert. Normalerweise zeigt nach Ausführung des Log-on bzw. des QPDEV Kommandos der SCI folgenden Kommando-Prompt am Bildschirm an:

SCI V 200 ()

Gibt der Benutzer das SPDEV Kommando (Start Program Development) ein, zeigt der SCI den folgenden Kommando-Prompt am Bildschirm an:

PDEV V 200 ()

Auf diese Weise kann der Benutzer leicht erkennen, ob er spezielle Program Development Merkmale des SCI verwenden kann oder nicht.

2.3.3 Kommandoprozedur

Ein in der SCI Sprache geschriebenes Programm, das ein SCI Kommando implementiert, wird "Kommandoprozedur" genannt. Es folgt eine Beispiel einer Kommandoprozedur für das SF Kommando:

Zeilennur.	Daten von der Plattendatei
1	.PROC SF(SHOW FILE)
2	FILE PATHNAME = ACNM (@SFP)
3	.SHOW &FILE PATHNAME
4	.EOP

2.3.4 Anfangswerte

Im obigen Beispiel ist der Wert des Promptfeldes FILE PATHNAME auf den Wert des Synonyms SFP gesetzt. Wenn die Prozedur ausgeführt wird, zeigt der SCI den Wert des Synonyms SFP als Anfangswert für FILE PATHNAME an. Hat das Synonym den Wert Null bzw. ist für das Synonym kein Klammerausdruck angegeben, dann hat FILE PATHNAME den Wert Blank, wenn SF ausgeführt wird.

2.3.5 Sonderzeichen

Bestimmte Zeichen haben eine besondere Bedeutung innerhalb einer Kommandoprozedur. Diese sind:

, Das Komma deutet auf eine Forsetzung einer Kommandozeile hin. Das Komma in der Zeile 1 des Beispiels besagt, daß der SCI das Promptfeld in der nächsten Zeile finden kann.

. Der Punkt wird allen SCI Primitives vorangestellt. .SHOW identifiziert das SHOW Primitive. SHOW ohne Punkt würde vom SCI nicht als Primitive erkannt werden.

& Das Und-Zeichen ermöglicht es dem SCI, zwischen Eingabeanforderungen einerseits, und Synonymen und Literalen andererseits zu unterscheiden. Nachdem sie definiert wurden, müssen Promptnamen ein Und-Zeichen vorangestellt werden.

@ Das kaufmännische At-Zeichen identifiziert Synonyme und deutet auf indirekte Referenzen hin. Der Ausdruck @SFP im Beispiel nimmt Bezug auf den Wert des Synonyms @SFP. Das At-Zeichen muß dem Namen des Synonyms vorangestellt werden, um es dem SCI zu ermöglichen es als Synonym identifizieren zu können. Ein weiteres Beispiel für den Gebrauch des At-Zeichens ist @&FILE PATHNAME. Hier wird der Wert des Prompts &FILE PATHNAME als Synonymnamen benutzt. Die Synonyms werden in einem anderen Teil dieses Kapitels näher beschrieben.

* Der Stern hat zwei Funktionen. Ein Stern in Spalte 1 einer Prozedurzeile identifiziert diese als Bemerkungszeile. Der SCI ignoriert solche Zeilen, wenn die Prozedur aufgerufen wird. Ein Stern, der dem Promttyp in einer Prompts-Definitions-Zeile vorangestellt wird, deutet auf einen optionalen Prompt hin. Zum Beispiel:

  FILE PATHNAME = *ACNM,

  In diesem Promptfeld darf der Benutzer die RETURN-Taste ohne vorherige Eingabe drücken. Fehlt der Stern, dann verlangt der SCI eine Eingabe in dieses Feld.

! Das Ausrufezeichen wird innerhalb eines SCI Kommandos benutzt, um eine Bemerkung einzuleiten. Hier ein Beipiel:

  SF(SHOW FILE), !Diese Prozedur zeigt den Inhalt einer Datei an

  Der SCI ignoriert eine dem Ausrufezeichen folgende Bemerkung. Der Gebrauch von Bemerkungen in SCI-Prozeduren wird für Protokollierungszwecke empfohlen.

" Zeichenketten, die als nicht eindeutig gelten, weil sie Sonderzeichen enthalten, oder wegen des Zusammenhangs in dem sie benutzt werden, müssen in Anführungszeichen eingeschlossen sein. Zum Beispiel:

"CHOOSE 1, 2, 3" = INT

Dieser Prompt beinhaltet auch ein Komma. Normalerweise interpretiert der SCI dieses Sonderzeichen als das Ende des Promptnamens. Im obigen Beispiel jedoch ist die Zeichenkette in Anführungszeichen eingeschlossen. Daher wird das Sonderzeichen als Teil des Promptnamens aufgefasst.

2.3.6 Schlüsselwort (Keyword)

In einer Anzahl Fehlermeldungen wird das Wort "Keyword" d.h., Schlüsselwort anstelle von "Prompt" benutzt. Diese zwei Ausdrücke sind identisch.

2.3.7 Prozedurnamen

Benutzerprozeduren können in einer Prozedurbibliothek gespeichert werden, indem der Directory-Name der Kommandobibliothek dem Namen der Prozedur vorangestellt wird, um so einen Pfadnamen zu bilden. Die restlichen in diesem Kapitel angegebenen Beispiele für Prozeduren benutzen den Prozedurnamen EXP. Diese Beispiele können in der System-Prozedurbibliothek gespeichert werden, indem der Name .S$PROC.EXP als Name der Ausgabedatei im Text-Editor angegeben wird.

MERKE

Der SCI überprüft den Prozedurnamen in der ersten Prozedurzeile und den Namen .S$PROC im Pfadnamen der Datei auf Gleichheit. Ist die Prozedur "SF" unter dem Pfadnamen .S$PROC.EXP gespeichert, meldet der SCI Fehler FF02 und .S$PROC bricht die Prozedur ab, wenn EXP aufgerufen wird.

Prozeduren können nicht nur in der Systemprozedur-Bibliothek gespeichert werden, sondern auch unter jedem Directory des TAXO-Systems. Durch Benutzung der .USE Primitive, die in diesem Kapitel beschrieben wird, kann der SCI veranlaßt werden, nicht .S$PROC, sondern eine anderes Directory anzusprechen. .USE erlaubt es dem Benutzer die den Prozedurnamen voranzustellenden Directories zu spezifizieren, um so einen Pfadnamen zu bilden. Der SCI kann dazu veranlaßt werden, eine Prozedur namens DROP, die in einer Datei namens SYS1.JF0012.DROP gespeichert ist, zu suchen. Der Benutzer führt das folgende Primitive-Kommando aus:

.USE .S$PROC, SYS1.JF0012

Dieses Kommando kann als Antwort auf den SCI-Prompt eingegeben werden.

Der SCI durchsucht zuerst das Directory .S$PROC, wenn DROP als SCI Kommando eingegeben wird; ist diese Datei nicht vorhanden, durchsucht der SCI anschließend das Directory SYS1.JF0012. Fehlermeldung 900E UNKNOWN COMMAND NAME wird am Bildschirm angezeigt, wenn die Prozedur in keinem der beiden Directories gefunden werden konnte.

2.3.8 Kommando-Prozessor

Ein Kommando-Prozessor ist ein von einer Kommandoprozedur ausgeführter Task, der die Zielfunktion eines Kommandos vervollständigt. Kommando-Prozessoren können vom Benutzer geschriebene Programme oder Tasks sein oder auch Dienstprogramme.

2.3.9 Primitive-Kommandos

Die Grundoperatoren der SCI Sprache werden Primitives genannt. Im Beispiel für SF ist .SHOW ein Primitive. Es ruft einen SCI-Prozessor, der dann den Inhalt der angegebenen Datei am Bildschirm anzeigt. Es gibt weitere Primitives für logische Operationen (.LF, .ELSE, .ENDIF) für die Manipulation von Daten (.SHOW, .SPLIT) für Steuerungszwecke (.OPTION, .STOP) usw. Innerhalb einer Prozedur werden Primitives durch einen dem Namen vorangestellten Punkt identifiziert.

2.3.10 Vordergrund-Modus (Foreground Mode)

Der Foreground Modus ist eine Methode der Programmausführung die den SCI am Benutzer-Terminal unterbricht. In diesem Modus kann eine Task das Terminal für die Ausgabe von Prompts oder für die Eingabe oder Ausgabe anderer Daten benutzen. Während der Ausführung einer Task im Vorderung werden weder SCI Kommandos noch andere Benutzeranforderungen vom Terminal entgegengenommen.

2.3.11 Hintergrund-Modus (Background-Mode)

Der Background Modus ist eine Methode der Programmausführung in der ein Task, unabhängig vom Benutzer-Terminal ausgeführt wird. Während der Ausführung eines Tasks im Hintergrund kann der Benutzer andere Prozesse im Vordergrund-Modus zur Ausführung bringen. Ein Task der im Hintergrund-Modues ausgeführt wird, kann auf das Terminal nicht zugreifen, ausgenommen auf die Meldungen, die in das TLF geschrieben werden.

Im Background-Modus wird eine Kopie der zum Zeitpunkt der Aktivierung der Task verfügbaren Synonymtabelle der im Hintergrund ausführenden Task zur Verfügung gestellt. Abänderungen von Synonymen, die nach der Aktivierung der im Hintergrund ausführenden Task, im Vordergrund vorgenommen werden, haben auf die der im Hintergrund ausführenden Tasks zur Verfügung gestellten Synonyme keinen Einfluß und umgekehrt.

2.3.12 .BID, .QBID, .DBID

Wenn eine Prozedur einen Kommandoprozessor aufruft, wird dies "Bidding the Processor" genannt. .BID aktiviert einen Task im Vordergrund-Modus. .QBID aktiviert einen Task im Hintergrund. .DBID aktiviert einen Hintergrund-Task im Debug-Modus.

2.3.13 Stapelverarbeitung (Batch Stream)

Ein Batch-Stream ist eine in einer Benutzerdatei gespeicherten Gruppe von Kommandos, die eine Folge von Tasks oder Kommandos aktiviert. Ein Batch-Stream läuft im Hintergrund ab. Aus diesem Grund müssen alle notwendigen Benutzerantworten im Batch vorhanden sein, da ein im Hintergrund ausführender Task mit dem Terminal nicht kommunizieren kann.

2.3.14 TLF (Terminal Local File)

Das Terminal Local File kurz TLF genannt, wird dazu benutzt um Information von einem Kommandoprozessor an den Benutzer weiterzugeben. Jedes Terminal verfügt über zwei solche Dateien, eine für die Vordergrund-Prozessoren und eine für die Hintergrund-Prozessoren. Wenn eine im Vordergrund ausführende Prozedur beendet wird, wird das Vordergrund TLF vom SCI auf evtl. vorhandene Daten überprüft und diese am Terminal ausgegeben.

Wenn ein im Hintergrund ausführender Task beendet wird, wird der Inhalt des Hintergrund TLF's nicht automatisch angezeigt. Der Benutzer kann eine Anzeige des TLF veranlassen, indem er das SBS (Show Background Status) bzw. das WAIT-Kommando aufruft.

Das TLF wird stets in "Extended-Modus" eröffnet, wenn ein Task darauf zugreift. Ruft der Benutzer weder das SBS nocht das WAIT-Kommando auf, werden die im Hintergrund TLF gespeicherten Meldungen weder angezeigt noch werden sie gelöscht. Wird der Inhalt des TLF dann doch zur Anzeige gebracht, so erscheinen zuerst die alten Meldungen. Da manche Routinen, z.B. der Link Editor das TLF für die Ausgabe von Fehlermeldungen benutzen, wird es empfohlen, SBS bzw. WAIT nach jeder Ausführung einer Prozedur im Hintergrund aufzurufen, um Verwirrungen zu vermeiden.

Ein Prozessor kann auf das TLF mittels S$OPEN, S$WRT und S$CLOS zugreifen. Diese Unterprogramme sind im "TAXO SVC und S$ Interface" näher beschrieben.

2.3.15 Terminal Communication Area

Die Terminal Communication Area (TCA) ist eine vom SCI, den Kommandoprozeduren und den Kommandoprozessoren benutzte Datenstruktur. Sie besteht in der Hauptsache aus einer Name Correspondence Table (NCT). Dieses ist eine Tabelle, welche die Synonyme und ihre Werte enthält. Die TCA wird auch zur Speicherung von Meldungen benutzt, die vom Vordergrund SCI am Terminal angezeigt werden.

2.3.16 Privilege Levels

Ein Privilege Level (Berechtigungsstufe) wird einem Benutzer zusammen mit seiner User-ID zugewiesen. Den SCI-Prozeduren können Privilege Levels zugeordnet werden, die sie gegen Zugriff durch Unberechtigte schützen. Die Standard-Privilege-Levels sind in einem weiteren Teil dieses Kapitels näher beschrieben.

2.3.17 Menüs

Eine zusätzliche Funktion des SCI ist das Zugreifen auf und das Anzeigen von Menüs. Menüs sind sequentielle Textdateien, die mittels Text-Editor editiert werden können. Ein Menü kann eine Liste von Kommandos sein, die dem Benutzer helfen das System zu steuern, oder jegliche andere Art von Information, die dem Benutzer Unterstützung gibt. Jedesmal, wenn das System in den Kommando-Modus zurückkehrt, versucht der SCI das gegenwärtige Menü am Bildschirm anzuzeigen.

Ein Menü kann aus bis zu 23 Zeilen bestehen. Die Primitives .MENU, .USE und .OPTION beeinflussen die Wirkungsweise des SCI in Bezug auf Menüs.

Das SF Kommando (Show File) kann benutzt werden, den Inhalt des Standardmenüs am Bildschirm anzuzeigen, durch Eingabe des Pfadnamens .S$PROC.M$LC. Der Bildschirm zeigt das für den Aufbau von Menüs notwendige Format. Das Verfahren für das Ändern des vorgegebenen Menüs wird in Kapitel 3. beschrieben.

3 STANDARD-DATEN

Die vom SCI benutzten Standard-Dateien sind:

- .S$NEWS - die Nachrichtendatei (News File)
- .S$PROC.M$00 - die Datei für die Log-on Prozedur
- .S$PROC.M$01 - die Datei für die Log-off Prozedur
- .S$PROC.M$LC - das Vorgabemenü für .S$PROC

Benutzer können auf jede dieser Dateien zugreifen, und sie mittels Text-Editor modifizieren. Vorgabe-Menüdateien S$PROC.M$LC, die .S$PROC.M$00 Datei, die die Log-on Prozedur enthält und Datei .S$PROC.Ms01, die die Log-off Prozeduren enthalten, sind Teil des TAXO Standard-Software. Die Datei .S$NEWS kann nach Bedarf vom Benutzer erstellt werden.

3.1 Die Datei .S$NEWS

Der Inhalt dieser Datei wird vom SCI nach Beendigung der Log-on Prozedur angezeigt. Alle Informationen, die den Benutzern eine Hilfe sein können, können in dieser Datei abgespeichert werden.

Nach Beendigung des Log-on an einem Terminal sucht der SCI eine Datei namens .S$NEWS. Wird sie gefunden, zeigt der SCI den Inhalt dieser Datei vor Ausgabe des Vorgabemenüs und des SCI-Prompts am Bildschirm an. Die Datei kann auch dazu benutzt werden, an den Benutzern während des Log-on Information weiterzugeben. Z.B., wurde eine neue Kommandoprozedur inzwischen in die Bibliothek der Systemprozeduren übernommen, kann die Datei .S$NEWS dazu benutzt werden, die Benutzer über das Vorhandensein dieser Kommandoprozedur in Kenntnis zu setzen.

3.2 .S$PROC.M$00 und .S$PROC.M$01

Die Dateien .S$PROC.M$00 und .S$PROC.M$01 werden als Prozeduren vom SCI zur Zeit des Log-on bzw. Log-off aufgerufen. Der Benutzer kann diese Prozeduren so programmieren, daß sie zum Ändern des SCI-Kommandoprompts, zum Einfügen einer Bibliothek für Benutzerprozeduren oder zu irgend einem anderen Zweck benutzt werden können. Ein Beispiel für die Standard-Datei M$00 wird zusammen mit der Beschreibung der .USE Primitive in einem anderen Teil dieses Handbuches aufgeführt.

Datei M$01 wird vom SCI als Prozedur aufgerufen wenn ein Benutzer sich durch Eingabe des Q Kommandos (Quit) abmeldet. Dadurch kann der Benutzer etwaige andere für die Abmeldung notwendigen Tasks aufrufen.

3.3 Menüdateien - .S$PROC.M$LC

Die Datei .S$PROC.M$LC enthält das vom System vorgegebene Menü. Diese Dateien können mittels Text-Editor modifiziert werden, um die von Benutzern geschriebenen Prozeduren und Menüs einzubeziehen.

Die Primitives .OPTION, .MENU und .USE können die Wirkungsweise der Menüfunktion beeinflußen.

Das vom System vorgegebene Menü kann nur dann geändert werden, wenn sämtliche Terminals mit Ausnahme des Terminals, das die Änderungen ausführen will, abgemeldet sind und wenn das Terminal das die Änderung durchführen will, das Standardmenü nicht benötigt. Ein LUNO ist dem Vorgabemenü zugewiesen. Wird diese von irgendeinem Terminal gebraucht, kann die Datei nicht geändert werden. Das Terminal, welches die Menüdatei editiert, kann durch Benutzung der Primitive .OPTION, MENU = Parameter, einer anderen Menüdatei zugeordnet werden. Dadurch wird die dem Vorgabemenü zugeordnete LUNO freigegeben und das Standardmenü kann mittels Text-Editor editiert werden.

Folgende Schritte werden zur Änderung des Standardmenüs .S$PROC.M$LC benötigt, wenn angenommen wird, daß M$LC das Standardmenü ist und .S$PROC dieProzedurbibliothek:

1. Alle Benutzer müssen sich abmelden.

2. Ein Terminal muß angemeldet werden.

3. Als Antwort auf den SCI-Prompt muß SPDEV eingegeben werden. Primitives und Programm Development Kommandos sind jetzt verfügbar.

4. .OPTION MENU=MYMENU als Antwort auf den SCI-Prompt eingeben. MYMENU kann eine beliebige Menüdatei in Bibliothek .S$PROC sein.

MERKE

Die Primitive .OPTION stellt dem Menünamen, der als Parameter in der MENU-Option angegeben wurde, automatisch $M voran. Daher adressiert MYMENU in Wirklichkeit .S$PROC.M$MYMENU.

5. Menüdatei .S$PROC.M$LC editieren.

6. Nach Abschluß des Editierens die Primitive .OPTION benutzen, um das Standardmenü wieder auf M$LC zurückzuschalten (.OPTION MENU = LC).

7. Das System kann nun ganz normal weiter benutzt werden.

4 DIE SCI SPRACHE

Die SCI Sprache ist eine Programmiersprache. Sie besteht aus Kommandos und Datendefinitions-Anweisungen, die dazu benutzt werden, die Arbeitsweise von TAXO im Bezug auf die vom Benutzer eingegebenen Prozedurnamen zu steuern. Die Komponente dieser Sprache sind die Prompts, die Synonyme und die Primitives.

4.1 Prompts (Eingabeanforderungen)

Promptvariable können zu dem Zeitpunkt, zu dem der jeweilige Prompt deklariert wird einem Typ zugeordnet werden. Eine Liste der zulässigen Typenbezeichnungen kann in Tabelle 4-1 nachgelesen werden.

Tabelle 4-1	Zulässige Typenbezeichnungen für Prompts
ACNM [(Anfangswert)]	Der Pfadname einer Datei oder Gerätename eines Geräts
INT [(Anfangswert)]	Eine Ganzzahl
NAME [(Anfangswert)]	A bis Z, 0 bis 9
STRING [(Anfangswert)]	Alphanumerisch, einschl. Sonderzeichen
YESNO [(Anfangswert)]	Benutzerantwort Y (yes) oder N (no)

Alle Promptypen haben die folgenden gemeinsamen Eigenschaften:

1. Ein Stern (*) vor der Typenbezeichnung bedeutet, daß eine Benutzerantwort optional ist. Der Benutzer kann die RETURN-Taste ohne vorherige Eingabe eines Wertes drücken.

2. Jedem Promptyp kann ein Anfangswert zugewiesen werden.

3. Anfangswerte, die bis auf eine mit einem Dollar-Zeichen beginnende Zeichenkette aufgelöst werden können, lösen sich letztendlich in eine Null-Zeichenkette auf. Dies wird im Kapitel "Das Dollar-Zeichen innerhalb eines Prompts" näher erläutert.

4. Einem Prompt kann ein einziger Wert oder eine Wertliste zugewiesen werden. Um einem Prompt eine Wertliste zuzuweisen, muß der Promptyp in Klammern eingeschlossen sein. Zum Beispiel, FILE = ACNM ordnet dem Prompt FILE den Typ ACNM zu. INPUT = (ACNM) bedeutet, daß INPUT entweder ein einfacher Wert sein kann oder eine Wertliste vom Typ ACNM. Eine Liste wird als Antwort auf einen Prompt eingegeben, die einzelnen Werte in der Liste durch Komma voneinander getrennt.

5. Anfangswerte, deren Wert aus einer Liste besteht, müssen in Anführungsstrichen eingeschlossen sein.

Die folgende Eingabe ist zulässig als Antwort auf den Prompt INPUT = (ACNM):

MY.INP.F1,MY.INP.F"

INPUT beinhaltet eine aus zwei Elementen bestehende Liste: MY.INP.F1 und MY.INP.F2. Diese können mit dem Primitiv Kommando .SPLIT getrennt werden.

MERKE

Die Elemente müssen in Klammern stehen wenn die Liste mittels eines Batch Streams eingegeben wird. Die Klammern können weggelassen werden wenn die Liste aus nur einem Element besteht.

Promptnamen dürfen die folgenden Sonderzeichen nicht enthalten:

@	Das kaufmännische At-Zeichen
	Das Fehlzeichen
=	Das Gleichheitszeichen
&	Das Und-Zeichen
"	Anführungsstriche
'	Der Apostroph
,	Das Komma

Die Zeichen Plus (+), Minus (-), Schrägstrich (/) und Stern (*) beinflußen das Primitiv Kommando .EVAL und sollten in Promptnamen nicht benutzt werden.

Ein Promptname kann aus bis zu 28 Zeichen, einschließlich Leerstellen, bestehen. Die Antwort auf einen Prompt kann bis zu 50 Zeichen lang sein.

Die folgenden Unterkapitel beschreiben jeden Promptttyp und sein Format.

4.1.1 Promptttyp ACNM

Promptttyp ACNM verlangt Eingabe eines Dateipfadnamens. ACNM ist die Abkürzung für "Access Name". Das folgende Beispiel zeigt ACNM als Promptttyp:

FILE PATHNAME = *ACNM(VALUE)

Im obigen Beispiel zeigt der dem Promptttyp vorangestellte Stern, daß die Benutzerantwort optional ist. Da ACNM nicht im Klammern eingeschlossen ist, darf die Benutzerantwort nur aus einem einfachen Wert bestehen. Die die Zeichenkette VALUE umschliessende Klammer bedeutet, daß diese Zeichenkette als Anfangswert benutzt wird.

4.1.2 Promptttyp INT

Dieser Promptttyp muß eine aus 32 hexadezimalen oder dezimalen Ziffern bestehende Ganzzahl zwischen> 80000000 und >7FFFFFFF, bzw. -2147483648 und 2147483647 sein. Ein > Zeichen, bzw. eine führende Null in der Eingabezeichenkette kennzeichnet eine hexadezimale Konstante. Promptttyp INT wird im folgenden Beispiel gezeigt:

PARM1=INT(0)

Im obigen Beispiel kennzeichnen die den Wert umschließenden Klammern (0) den Anfangswert.

4.1.3 Promptyp NAME

Dieser Promptyp muß eine mit einem Dollar-Zeichen ($) oder einem Buchstaben (A-Z) beginnende Zeichenkette sein; die Zeichenkette darf nur Großbuchstaben (aus der englischen Sprache), das Dollar-Zeichen oder Ziffern zwischen 0 und 9 enthalten. Es folgt ein Beispiel dieses Prototyps:

TASK NAME = *NAME

Im obigen Beispiel bedeutet der dem Prototyp *NAME vorangestellte Stern, daß die Benutzerantwort optional ist.

4.1.4 Promptyp STRING

Dieser Promptyp wird als eine Zeichenkette definiert, die folgende Zeichen nicht enthalten darf: Anführungsstriche, Ausrufezeichen, Gleichheitszeichen, Klammern und Komma-Zeichen. STRING darf aber auch eine in Anführungsstriche eingeschlossene Zeichenkette sein, genannt "Quoted String". Eine solche Zeichenkette darf Anführungsstriche, Leerstellen, Ausrufezeichen, Gleichheitszeichen, Klammern und Komma-Zeichen enthalten.
Innerhalb eines "Quoted String" angegebene Anführungsstriche müssen als doppelte Anführungsstriche geschrieben werden, d.h. "". Paare von Anführungsstrichen innerhalb eines "Quoted String" werden vom SCI als einfache Anführungsstriche interpretiert:

"PRESS ""RETURN"" TO CONTINUE"

wird vom SCI als PRESS "RETURN" TO CONTINUE interpretiert.

Hier ein Beispiel für Promptyp STRING:

TEXT = *(STRING) (@XES)

Dieses Beispiel zeigt folgendes:

1. Die den Promptyp umschließenden Klammern bedeuten, daß die Benutzerantwort entweder ein einfacher Wert oder eine Wertliste sein darf.

2. Der dem Promptyp vorangestellte Stern (*) bedeutet, daß eine Benutzerantwort optional ist.

3. Das den Zeichen XES vorangestellte At-Zeichen (@) bedeutet, daß der Wert des Synonyms XES, falls vorhanden, als Anfangswert genommen wird.

4.1.5 Promptyp YESNO

Dieser Promptyp besteht aus einer alphabetischen Zeichenkette, die entweder mit Y oder N beginnen muß. Es folgt eine Beispiel dieses Promptyps:

ARE YOU SURE?=YESNO

In diesem Beispiel muß die Benutzerantwort mit Y oder N beginnen. YESSIR ist eine zulässige YES-Antwort, und NOSIR eine zulässige NO-Antwort. SCI wertet nur das erste Zeichen der Antwort aus.

In einem Vergleich mittels .IF oder irgend eines anderen Vergleichs-Primitiv-Kommandos vergleicht der SCI alle Zeichen der beiden Operanden. Das Ergebnis eines Vergleichs zwischen Zeichenkette A und Zeichenkette B, indem A einen Wert von YES und B einen Wert von YESSIR hat, ist "größer als", da A weniger Zeichen enthält. Aus diesem Grund sollten Prozeduren, die diesen Promptтyp benutzer, Techniken anwenden, welche: (1) die Benutzerantwort auf Zulässigkeit überprüfen und die gegenteilige Antwort automatisch annehmen oder (2) die Benutzerantwort auf Zulässigkeit überprüfen und die Prozedur bei Unzulässigkeit abbrechen. Die Beschreibung des Primitiv-Kommandos .IF erläutert diese Techniken.

4.1.6 Abkürzungen für Prompts

Da Promptnamen mitunter etwas lang sind, erlaubt der SCI die Anwendung von Abkürzungen. Die Benutzung von Abkürzungen ist besonders nützlich für das Schreiben von Batches. Der für das Vergleichen abgekürzter Prompts angewendete Algorithmus ist in den folgenden Abschnitten beschrieben.

Der Prompt für das erste Feld einer Prozedur könnte beispielsweise SOURCE ACCESS NAME sein und seine Abkürzung SO. Der angewendete Algorithmus arbeitet wie folgt:

1. Das erste Zeichen des Prompts (SOURCE ACCESS NAME) muß mit dem ersten Zeichen seiner Abkürzung (SO) identisch sein.

2. Sämtliche im abgekürzten Prompt vorkommende Zeichen müssen im vollen Promptnamen in der gleichen Reihenfolge vorhanden sein.

3. Der abgekürzte Prompt muß alle, in dem vollen Promptnamen vorhandene numerische Zeichen ebenfalls aufweisen. Zum Beispiel: ist der volle Promptname SOURCE1 ACCESS NAME, so muß die Abkürzung mindestens S1 sein.

4. Das erste, im abgekürzten Prompt angegebene Zeichen, das nicht dem aktuellen Wort des vollen Promptnamens entnommen wurde, muß dem nächsten Wort des vollen Namens entnommen worden sein. Der Prompt SOURCE ACCESS NAME könnte mit SOA (die ersten zwei Zeichen aus SOURCE und das erste Zeichen aus ACCESS), aber nicht mit SON abgekürzt werden.

5. Eine Abkürzung muß eindeutig sein und darf für einen anderen Prompt in der gleichen Prozedur nicht auch die gültige Abkürzung darstellen. DELETE THE FILE? und DATE TP PRINT könnten zwar beide mit DT abgekürzt werden, doch könnte der SCI den gewünschten Prompt nicht eindeutig identifizieren. In diesem Fall wären DTF (für DELETE THE FILE?) und DTP (DATE TO PRINT) die korrekten Abkürzungen.

4.1.7 Synonyme

Synonyme sind benannte Zeichenketten, die als Teil des mit der User-ID verbundenen Workspace gespeichert werden. Synonyme werden häufig benutzt, um langen Dateinamen eine kürzere Bezeichnung zuordnen zu können. Sie können auch dazu benutzt werden, Zeichenketten die einer Prozedur oder einem Programm verfügbar gemacht werden müssen, zu speichern. Wurde ein Synonym einmal für eine User-ID definiert, so steht es allen unter dieser ID ausgeführten Programmen und Prozeduren zur Verfügung. Auf diese Weise dienen Synonyme der Kommunikation zwischen Prozeduren und Programmen, Prozeduren und anderen Prozeduren, usw.

Statt eine ganze Zeichenkette für einen Dateinamen einzugeben kann der Benutzer dem Wert des Dateinamens ein Synonym mittels dem AS Kommando (Assign Synonym) zuweisen. Zum Beispiel:

```
( ) AS                                          !Kommando eingeben
ASSIGN SYNONYM VALUE                            !Die SCI Prompts werden ausgegeben
            SYNONYM:   MYFILE                   !Felder eingeben
              VALUE:   .S$PROC.EXP
```

Nach Ausführung dieses Kommandos kann Datei .S$PROC.EXP mit MYFILE angesprochen werden. Auch darf jede Kommandoprozedur dieses Synonym benutzer mittels der Angabe @MYFILE. Z.B. folgendes eingeben, um Datei .S$PROC.EXP anzuzeigen:

```
.SHOW.@MYFILE
```

Die Zugänglichkeit der Synonyme verleiht ihnen eine große Nützlichkeit. Ein Verwendungszweck für Synonyme ist das Übertragen von Parametern von einer Prozedur zur anderen. Das Synonym SFP ist ein Standardsynonym, das immer auf den zuletzt ausgedruckten bzw. angezeigten Dateinamen Bezug nimmt. Dieses ist ein Standardsynonym in TAXO, das von einer Anzahl Systemkommandoprozeduren benutzt wird. Jede Prozedur und jeder Prozessor kann auf dieses Synonym zugreifen, oder ihm einen neuen Wert zuweisen. Die dafür vorgesehenen Techniken sind weiter hinten näher beschrieben.

Eine Anzahl der unter TAXO verfügbaren Standardsynonyme sind in Tabelle 4-2 ausgelistet. In unserem Beispiel für SF, veranlasst Synonym SFP das Zuordnen der zuletzt ausgedruckten bzw. angezeigten Datei als Anfangswert für den Prompt FILE PATHNAME. Das tatsächliche Aussehen der SF-Prozedur ist:

```
SF (SHOW FILE),       FILE PATHNAME = ACNM
   FILE PATHNAME = ACNM (@$SF$P)
   .SYN $SF$P = "&FILE PATHNAME"
   .IF"&FILE PATHNAME", NE, ""
   .SHOW @ &FILE PATHNAME
   .ENDIF
```

In dieser Form aufgerufen, weist das SF Kommando dem Synonym SFP einen Wert zu, der mit der als Antwort auf den Prompt FILE PATHNAME eingegebenen Zeichenkette identisch ist. Jede Prozedur, die das Synonym SFP als Anfangswert benutzt, benutzt automatisch den vom SF in Synonym SFP zuletzt geladenen Dateinamen, bzw. die Prozedur oder den Prozessor, der den Wert von SFP zuletzt upgedated hat. Das im Beispiel gezeigte Primitiv-Kommando .SYN ändert den Wert vom SFP auf die als Antwort auf den Prompt FILE PATHNAME eingegebene Zeichenkette.

Die Angabe@ FILE PATHNAME veranlasst den SCI, die Angabe@ FILE PATHNAME als Synonym statt als Datenfeld zu betrachten. Wird MYFILE als Antwort auf den Prompt FILE PATHNAME eingegeben, dann wird die Datei .S$PROC.EXP, falls dieser Wert dem Synonym MYFILE zugewiesen wurde, am Bildschirm angezeigt. Wird statt eines Synonymnamens ein Dateiname eingegeben, versucht der SCI zunächst, ein Synonym dieses Namens zu finden. Findet er kein Synonym mit diesem Namen, dann versucht er, eine Datei mit diesem Namen zu finden.

Kann der SCI einen, als Antwort auf einen Prompt eingegebenen Synonymnamen nicht auswerten, so nimmt er den, als Wert des Synonyms eingegebenen Namen. Dies ermöglicht die Eingabe von Synonymen sowie von Dateinamen als Antwort auf einen Prompt. Kann kein Synonym mit dem eingegebenen Namen gefunden werden, so wird der im Prompt eingegebene Name zum Anfangswert des Feldes gemacht.

Synonyme werden dadurch gelöscht, indem sie auf Null gesetzt werden. Dies wird innerhalb einer Prozedur mit folgender Angabe gemacht:

.SYN MYFILE = ""

Prozeduren, die Synonyme für den temporären Gebrauch erzeugen, müssen diese löschen, um einen Überlauf der Synonymtabelle zu vermeiden. Dies muß der letzte Schritt einer jeden Prozedur sein.

Der Benutzer kann den aktuellen Inhalt der Synonymtabelle durch Eingabe des LS Kommandos (List Synonyms) anzeigen lassen. Dieses Kommando zeigt sämtliche aktuelle Synonyme und ihre Werte am Bildschirm an. Tasten A (eine Zeile nach oben) und B (eine Zeile nach unten) ermöglichen es dem Benutzer, die ganze Liste zur Anzeige zu bringen.

Die folgende Tabelle zeigt eine Anzahl der, als Ergebnis der Ausführung einer normalen Prozedur vom SCI auf den neuesten Stand gebrachten Standardsynonyme.

Tabelle 4-2 Einige TAXO Standardsynonyme

$$MO	Ein 2-stelliger hexadezimaler Code für den SCI-Modus 00 = Batch Modus 01 = TTY Modus 0F = VDT Modus
$$ST	Eine 2-stellige dezimale Stationsnummer
$$UI	Eine 6-Zeichen lange Benutzer-Identifikation
ME	Ein 4-Zeichen langer Stationsname (z.B. ST09)
$$CC	Ein 5-stelliger hexadezimaler Code, der vom S$STOP im Vordergrund-Modus zurückgegeben wird
$$BC	Ein 5-stelliger hexadezimaler Code, der von einem in einer Hintergrund-Prozedur enthaltenen .STOP Primitiv-Kommando bzw. von einem in einer Hintergrund-Task enthaltenen S$STOP zurückgegeben wird

4.1.8 Auswertung von Synonymen

Synonyme werden wie folgt ausgewertet (die Auswertung wird durch @ eingeleitet): Nach Auffinden des ersten @Zeichens vergleicht der SCI den ersten Teil der nachfolgenden Zeichenkette vom dem @Zeichen bis zum ersten Sonderzeichen (d.h. bis zum ersten Zeichen, das weder ein alphanumerisches ASCII-Zeichen, ein numerisches ASCII-Zeichen noch ein Dollar-Zeichen ist), mit den in der Synonymtabelle gespeicherten Namen. Ungleichheit bedeutet, daß das Synonym undefiniert ist und ihm wird der Wert der Zeichenkette zugewiesen. In den Fällen, in welchen auf das Synonym kein Sonderzeichen folgt, kann das Synonym in Fehlzeichen (^) eingeschlossen werden, um eine korrekte Substitution zu ermöglichen. Zum Beispiel: wenn angenommen wird, daß der Name USER den Synonymwert ID01 hat und daß dieser Name im Text USERFILE vorkommt, ohne daß auf das Wort USER ein Sonderzeichen folgt, so müßte der Benutzer folgendes schreiben, um eine korrekte Synonymsubstitution vornehmen zu können:

@^USER^FILE

Nach erfolgter Substitution würde das Feld wie folgt aussehen (die Fehlzeichen (^) wurden entfernt):

ID01FILE

Prozeduren dürfen in Pfadnamen eingebettete Synonyme enthalten. Die als Antwort auf einen Prompt notwendigen Benutzereingaben dürfen Sonderzeichen wie das Fehlzeichen (^) und das At-Zeichen (@) nicht enthalten. Auf Grund dieser Einschränkung können interaktive Benutzerantworten den vollen Leistungsumfang der Synonymauswertung nicht in Anspruch nehmen. Wenn es als Benutzerantwort auf einen Prompt eingegeben werden würde, so würde das obige Beispiel die Ausgabe einer Fehlermeldung zufolge haben. Innerhalb einer Prozedur wird .SHOW@ USER FILE als .SHOW ID01FILE ausgewertet werden, gemäß den obengenannten Regeln.

Der Gebrauch von Synonymen für die Darstellung von Pfadnamen kann wegen der speziellen Bedeutung von Sonderzeichen in der Synonymauswertung problematisch sein. Ein Synonym kann ohne Schwierigkeiten für die Darstellung eines gesamten Pfadnamens bzw. des ersten Komponenten eines Pfadnamens benutzt werden. Schwierigkeiten entstehen dann, wenn ein Synonym als Sekundärkomponente eines Pfadnamens benutzt wird (z.B.: VOL1.MYCAT.S, wo S ein Synonym mit dem Wert SOURCE ist), da das Synonym nicht richtig ausgewertet wird, wenn dem Pfadnamen ein @Zeichen vorangestellt wird (z.B. @VOL1.MYCAT.S). Synonyme können als Sekundärkomponenten eines Pfadnamens fungieren, wenn die @Zeichen auf der richtigen Stelle innerhalb der Zeichenkette geschrieben werden (z.B. "VOL1.MYCAT.@S" wird als VOL1.MYCAT.SOURCE interpretiert)

4.1.9 Benutzung des Dollar-Zeichens in Benutzereingaben

Ein Prompt hat den Wert Null wenn sein Anfangswert mit einem Dollar-Zeichen ($) beginnt. Zum Beispiel:

INPUT FILE = ACNM (@ $ABC)

Dieser Prompt hat als Anfangswert den Wert des Synonyms $ABC. Ist das Synonym $ABC eine Null, würde normalerweise der Anfangswert sich in Synonymnamen $ABC auflösen; da jedoch der Synonymname mit einem Dollar-Zeichen beginnt, ist der Anfangswert des Prompts Null.

Dies kann für den Benutzer von großer Wichtigkeit sein, denn wenn ein Synonym einen mit einem Dollar-Zeichen beginnenden Wert z.B. $BC hat, dann wird jeder von diesem Synonym geladener Anfangswert sich in eine Null auflösen. Zum Beispiel:

INPUT FILE = ACNM(@ $ABC)

Synonym $ABC hat den Wert $BC. Der Anfangswert von INPUT FILE wird vorerst in $BC aufgelöst, dann aber wegen dem führenden Dollar-Zeichen weiter in eine Null aufgelöst.

4.2 SCI Primitiv Kommandos

Die SCI Primitiv Kommandos stehen auf der niedrigsten Stufe der SCI Sprache. Tabelle 4-3 enthält eine Liste der verfügbaren Primitiv Kommandos und ihre Parameter. Auf jedes SCI Primitiv Kommando wird in den folgenden Abschnitten separat eingegangen.

Mit Ausnahme des Primitiv Kommandos .PROC sind im folgenden die Definitionen der Primitiv Kommandos in alphabetischer Reihenfolge angegeben, wobei die Beziehung stehenden Primitiv Kommandos in Gruppen zusammengefaßt sind. Die in [ ] eingeschlossenen Angaben sind optional. Die in Kleinbuchstaben aufgeführten Angaben deuten auf den Typ der für den Parameter notwendigen Antwort hin.

Tabelle 4-3 SCI Primitiv Kommandos

Primitiv Kommando	Parameter
.PROC .EOP	<Name>[<voller Name>)] [= <Ganzzahl>] [,<Prompt-Liste>]
.BID .QBID .DBID	TASK=<Name/Ganzzahl >[,LUNO=<Ganzzahl >][,CODE= <Ganzzahl >] [,PARMS=(<Zeichenkette,...., Zeichenkette>)]
.DATA .EOD	<acnm>[,EXTEND=<YES/NO>)][,SUBSTITUTION=>YES/NO<] [,REPLACE=<YES/NO<]
.EVAL .EXIT	<Name>=<Wert>
.IF .ELSE .ENDIF	<OP1>,<Beziehung>,<Op2>
.LOOP .UNTIL	<Op1>,<Beziehung>,<Op2>
.WHILE .REPEAT	<Op1>,<Beziehung>,<Op2>

.MENU	[<Men] -Name>]
.OPTION	[PROMPT=<Zeichenkette>[ ] ,MENU=<Name>]
	[,PRIMITIVES=<YES/NO>]
.OVLY	Für den Systemgebrauch reserviert
.PROMPT	[(<voller Name>)] [=<Ganzzahl>] [,<Prompt-Liste>]
.SHOW	<acnm>[,acnm,....>]
.SPLIT	LIST = (<Ganzzahl>,<Ganzzahl>. . . <Ganzzahl>),
	FIRST = <Synonym>
	[,REST = <Synonym>]
.STOP	[TEXT = <Ganzzahl>] [,CODE=<Ganzzahl>]
.SVC	Für den Systemgebrauch reserviert
.SYN	<Name>="<Wert>"
.USE	[<acnm 1>[,<acnm 2>]]

4.2.1 Primitiv Kommandos .PROC und .EOP

Eine SCI Prozedur beginnt mit der .PROC Anweisung und endet mit einer .EOP Anweisung. Dieses Primitiv Kommando kann benutzt werden, um eine Kommandoprozedur mittels eines Batch oder interaktiv am Terminal statt mittels des Text Editors in eine Kommandoprozedur-Bibliothek zu installieren.

Das Primitiv Kommando .PROC hat folgendes Format:

.PROC<Name>[(<voller Name>)] [=<Ganzzahl>] [,Prompt-Liste>]

Der Zwangsparameter "Name" definiert den Namen der Prozedur. Dieser Name kann bis zu 8 Zeichen lang sein und darf mit Ausnahme des Dollar-Zeichens keine Sonderzeichen enthalten. Ein Prozedurname darf weder mit einer Ziffer zwischen 0 und 9 noch mit einem Dollar-Zeichen beginnen. Dem Prozedurnamen darf ein optioneller, in runden Klammern eingeschlossener, "voller Name" folgen. Der "volle Name" wird am Bildschirm bei Aufruf der Prozedur angezeigt.

Das wahlfreie Feld, [=<Ganzzahl>] muß, falls angegeben, unmittelbar hinter "voller Name" stehen und bestimmt die Privilegstufe des Kommandos. Tabelle 4-4 zeigt die verschiedenen Privilegstufen. Die Privilegstufen von Kommandos haben die folgende Bedeutung: einem Anwender, der eine Priviligstufe hat, der kleiner ist als die, die einer bestimmten Kommandoprozedur zugeordnet ist, wird die Verwendung des betreffenden Kommandos verweigert. (Die Privilegstufe eines Benutzers wird durch Aufrufen der Prozedur Assign User ID (AUI) bzw. Modify User ID (MUI) festgelegt). Sonst können einerseits die SCI Kommandos entsprechend ihrer Mächtigkeit und/oder Gefährlichkeit und andererseits die Privilegstufen der Anwender entsprechend ihrer Vertrauenswürdigkeit und/oder ihres Kenntnisstandes hierarchisch gegliedert werden. Die vorgegebene Privilegstufe ist Null.

Tabelle 4-4 Privilegstufen der Kommandos

Stufe	Bedeutung
1	Vom Benutzer definiert (Niedrigste Stufe des Zugriffsprivilegs)
2	Vom Benutzer definiert
3	Vom Benutzer definiert
4	Zugriffsprivileg für allgemeine Systemdienste
5	Zugriffsprivileg für Systemverwaltungsdienste
6	Zugriffsprivileg für Systempflegedienste
7	Zugriffsprivileg für spezielle Wartungsdienste

Das wahlfreie Komma, das eine .PROC Zeile abschließt besagt, daß eine Prompt-Liste der Prozedurdefinitions-Anweisung folgt. Diese Liste hat die Form:

PROMPT [=TYPE]

PROMPT ist eine, den Promptnamen darstellende Zeichenkette und TYPE einer der in Tabelle 4-1 aufgeführten gültigen Typen. Der Prompt wird am Bildschirm nicht angezeigt, wenn [= TYPE] nicht angegeben wird. Wird mehr als eine Prompt eingegeben, so muß hinter den Typ- oder Wertangaben ein Komma geschrieben werden. Das Komma hinter der letzten Angabe in der Liste kann entfallen.

```
.PROC EXP(EXAMPLE PROC) = 0,
          INPUT FILE = ACNM,
          OUTPUT FILE = ACNM(INITIAL),
          NUMBER OF COPIES = INT(1)
          .....
.EOP
```

Das oben aufgeführte Beispiel verlangt vom Benutzer drei Eingaben: INPUT FILE, OUTPUT FILE und NUMBER OF COPIES. Da NUMBER OF COPIES die letzte Angabe ist erscheint hinter ihr kein Komma.

Maximal 22 Prompts können am Bildschirm angezeigt werden.

Es folgt ein Beispiel für Primitiv Kommandos .PROC und .EOP:

```
.PROC EXP(EXAMPLE PROC)=0,
          INPUT PATHNAME = ACNM
          SF FILE = &INPUT PATHNAME
.EOP
```

Diese Kommandoprozedur kann in die Kommandoprozedur-Bibliothek übernommen werden, indem sie als Benutzerantwort auf einen SCI Prompt eingegeben wird. Zum Beispiel:

1. .PROC EXP(EXAMPLE PROC)=0, eingeben und RETURN drücken. Das Komma hinter der Null darf nicht vergessen werden. Der SCI Prompt wird nach erfolgter Eingabe vom Bildschirm gelöscht.

2. INPUT PATHNAME = ACNM eingeben und RETURN drücken.

3. SF FILE = &INPUT PATHNAME eingeben und RETURN drücken.

4. .EOP eingeben und RETURN drücken. Nach dieser Eingabe wird der SCI Prompt erneut angezeigt.

5. Die Prozedur ist jetzt in der aktuellen Prozedurbibliothek gespeichert, z.B. .S$PROC.EXP.

Diese Kommandoprozedur kann jetzt durch Eingabe von EXP als Antwort auf den SCI Prompt aufgerufen werden.

4.2.2 Primitiv Kommandos .BID, .QBID und .DBID

.BID bewirkt die Ausführung eines TAXO Tasks als Vordergrund Task eines Terminals. Tasks, die mit .BID aufgerufen werden, benutzen Synonyme gemeinsam und werden seriell ausgeführt (d.h. der Benutzer darf keine weiteren Tasks am Terminal aufrufen, bis dieser Task beendet wird).

Das Primitiv Kommando .BID besitzt Operanden, die den Kommandoprozessor identifizieren und an ihn Parameter übergeben.

.BID hat folgendes Format:

```
.BID TASK = <Name/Ganzzahl>[ ,LUNO=<Ganzzahl>][ ,CODE =<Ganzzahl>]
[ ,PARMS=(<Zeichenkette,....,Zeichenkette>)]
```

Der Parameter TASK = Name/Ganzzahl muß eine in einer Programmdatei gespeicherten Task identifizieren. Der LUNO Parameter, falls angegeben, muß die der Programmdatei zugewiesenen globalen LUNOs spezifizieren. Die Vorbesetzung für die LUNO ist Programmdatei .S$PROGA. CODE ist eine Ganzzahl (zwischen 0 und 255), auf die der Task als Binärwert zugreifen kann. Die Vorbesetzung für CODE ist Null. PARMS ist eine Liste von Zeichenketten, die durch Kommas voneinander getrennt sind und auf die der Task zugreifen kann.

Durch .BID übergibt der SCI die Kontrolle an den Task. Wenn der Task beendet ist, führt der SCI die nächste Anweisung aus. Ein Beispiel für das Primitiv Kommando .BID:

```
.PROC EXP(EXAMPLE PROC - BIDS LIST DIRECTORY)=0,
      PATHNAME = *ACNM(@ $LD$P),
      LISTING ACCESS NAME = *ACNM
.SYN $LD$P = "&PATHNAME"
.IF "&PATHNAME",NE?""
.BID TASK = >32,CODE = >32,
      PARMS =(>32,@ &PATHNAME,@ &LISTING ACCESS NAME)
.ENDIF
.EOP
```

Das obige Beispiel zeigt den Gebrauch des Primitiv Kommandos .BID für die Ausführung einer, eine ID > 32 besitzende und in Programmdatei .S$PROGA gespeicherten Task.

Zwei der an diesen Task übergebenen Parameter sind Prompts. Literale und Synonyme dürfen ebenfalls als Parameter übergeben werden. Kommas werden benutzt um die Parameter voneinander zu trennen. Der aufgerufene Task benutzt Schnittstellenunterprogramme um auf die Parameter zuzugreifen.

Diese Unterprogramme sind in den Handbüchern der verschiedenen Programmiersprachen beschrieben.

Primitiv Kommandos .QBID und .DBID haben das gleiche Format wie .BID. .QBID bewirkt die Ausführung eines TAXO Tasks als Hintergrund Task. .DBID arbeitet in der gleichen Weise wie .QBID, mit dem Unterschied, daß der Hintergrund Task zwecks Austesten suspendiert bleibt.

4.2.3 Primitiv Kommandos .DATA und .EOD

Unter Verwendung des Primitiv Kommandos .DATA können Daten direkt von einem Eingabe-Batch in eine Datei einkopiert werden. Das Primitiv Kommando .DATA hat folgendes Format:

```
.DATA<acnm>[ ,EXTEND=<YES/NO>][ ,SUBSTITUTION=<YES/NO>]
[ ,REPLACE =<YES/NO>]
```

Der Datenstrom muß mit dem Primitiv Kommando .EOD abgeschlossen werden.

Die Daten werden in die Datei einkopiert, die mit acnm spezifiziert ist. Der Benutzer kann drei Parameter eingeben, um die Operation des Kopierens zu beinflussen. Diese drei Parameter sind EXTEND, SUBSTITUTION und REPLACE.

Der EXTEND Parameter wird dazu benutzt um zu spezifizieren, ob die Datei als erweiterte Datei geöffnet werden soll oder nicht. Der Parameter erlaubt es dem Benutzer, eine Anzahl Datenströme unter einem Pfadnamen zusammenzufassen. Die Vorbesetzung für EXTEND ist NO.

Der SUBSTITUTION Parameter gibt an, ob Texte im Datenstrom vor dem Kopieren auf die spezifizierte acnm ausgetauscht werden sollen. Wird YES angegeben, so werden die angegebenen Prompts und Synonyme gegen die entsprechenden Werte getauscht. Nicht in runden Klammern eingeschlossene mehrfache Leerstellen werden zu einer Leerstelle komprimiert. Die Vorbesetzung für SUBSTITUTION ist NO.

Der REPLACE Parameter gibt an, ob die durch acnm spezifizierte Datei, falls sie bereits existiert, durch den Datenstrom ersetzt werden soll. Die Vorbesetzung für REPLACE ist YES.

Das folgende Beispiel zeigt die Anwendung der Primitiv Kommandos .DATA und .EOD:

```
.PROC EXP(EXAMPLE PROC)=0,
          INPUT PATHNAME=ACNM(@ $EX$IP),
          OUTPUT PATHNAME=(ACNM)
          CC IAN =@ &INPUT PATHNAME,
             OAN =@ &OUTPUT PATHNAME
.DATA VOL1.MYLIB.MESSAGE,SUBSTITUTION=YES
COPY COMPLETED FOR: @ &INPUT PATHNAME
                  TO: @ OUTPUT PATHNAME
.EOD
.EOP
```

Im obigen Beispiel werden Primitiv Kommandos .DATA und .EOD für die Ausgaben der folgenden Meldung auf Datei VOL1.MYLIB.MESSAGE nach Kopieren einer Eingabedatei auf eine Ausgabedatei verwendet:

COPY COMPLETED FOR: "FILE NAME"
TO: "FILE NAME"

Das im Beispiel aufgeführte CC Kommando (Copy/Concatenate) kopiert die Eingabedatei auf die Ausgabedatei. Nach Ablauf dieser Operation wird in Datei VOL1.MYLIB.MESSAGE eine Meldung geschrieben, die die Namen der Eingabe- und Ausgabedateien enthält. IAN und OAN sind Promptnamen, die gemäß den bereits beschriebenen Regeln für Abkürzungen gebildet wurden. IAN ist die Abkürzung für INPUT ACCESS NAME(S) und OAN die Abkürzung für OUTPUT ACCESS NAME. Diese sind die von Kommandoprozedur CC benötigten Prompts.

Die Erläuterung des Primitiv Kommandos .IF enthält eine nähere Beschreibung über die Benutzung von Prozeduraufrufen innerhalb Prozeduren.

4.2.4 Primitiv Kommando .EVAL

Das Primitiv Kommando .EVAL hat folgendes Format:

.EVAL <Name>=<Wert>

.EVAL wertet die durch 'Wert' dargestellte Zeichenkette als numerischen Ausdruck, konvertiert das Ergebnis in Dezimal-ASCII und speichert ihn als Wert des durch NAME spezifizierten Synonyms. Das folgende Beispiel ordnet dem Synonym RESULT den Wert 13 zu.

```
.SYN THREE = 3
.SYN TWO = 2
.SYN RESULT = 0
.SYN RESULT = @ RESULT+@ THREE*5-@ TWO
```

Das Primitiv Kommando .EVAL stellt die arithmetischen Fähigkeiten des SCI zur Verfügung.

MERKE

Dem Synonym RESULT darf kein At-Zeichen (@) links des Gleichheitszeichens vorangestellt werden. Wird das gleiche Synonym rechts vom Gleichheitszeichen geschrieben, so muß ihm ein At-Zeichen (@) vorangestellt werden.

Werden Promptnamen in einem .EVAL Kommando geschrieben, so darf nur einer dieser Namen rechts vom Gleichheitszeichen erscheinen; darüberhinaus muß dieser der letzte Name in der Zeile sein. Zum Beispiel:

.EVAL RESULT = @ RESULT+&NO OF COPIES

Wird der Prompt &NO OF COPIES als erster Operand hinter dem Gleichheitszeichen geschrieben, dann wird das Plus-Zeichen (+) als Teil des Prompts interpretiert.

4.2.5 Primitiv Kommando .EXIT

Dieses Primitiv Kommando beendet die Ausführung einer Kommandoprozedur (das Primitiv Kommando .EOP beendet eine Kommandodefinition). .EXIT kann beliebig oft und an beliebigen Stellen in einer Kommandoprozedur verwendet werden.

Das Primitiv Kommando .EXIT hat folgendes Format:

```
.EXIT
```

Das folgende Beispiel zeigt die Anwendung dieses Primitiv Kommandos:

```
.PROC EXP(EXAMPLE PROC)=0,
          INPUT PATHNAME=ACNM(@ $EX$IP),
          DELETE FILE?=YESNO(N)
          SF FILE =&INPUT PATHNAME
      .IF &DELETE FILE?,NE,"Y"
      .EXIT
      .ENDIF
          DF PATHNAME = &INPUT PATHNAME
.EOP
```

In diesem Beispiel beendet das Primitiv Kommando .EXIT die Ausführung der Prozedur wenn &DELETE FILE? nicht gleich Y ist. Ist &DELETE FILE? gleich Y, dann wird die Eingabedatei gelöscht.

4.2.6 Primitiv Kommando .IF, .ELSE und .ENDIF

Das bedingte Primitiv Kommando der SCI Sprache entspricht der üblichen IF-THEN bzw. IF-THEN-ELSE Konstruktion. Das bedingte Primitiv Kommando .IF wird in Verbindung mit den bedingten Primitiv Kommandos .ELSE und .ENDIF benutzt. Primitiv Kommandos .IF und .ELSE erlauben es dem Benutzer, eine Aktion von dem Ergebnis eines Vergleichs abhängig zu machen. Das Primitiv Kommando .ENDIF beendet das Primitiv Kommando .IF.

Das Primitiv Kommando .IF hat folgendes Format und muß in Verbindung mit .ENDIF und u.U. auch mit .ELSE verwendet werden:

```
.IF       <Op1>,<Beziehung>,<Op2>
          :
.ELSE
          :
.ENDIF
```

Ist die durch .IF gestellte Bedingung erfüllt, so werden die dem .IF folgenden Anweisungen ausgeführt. Ist die Bedingung nicht erfüllt, dann werden die hinter .ELSE geschriebenen Anweisungen, falls .ELSE vorhanden ist, ausgeführt. Anschließend werden die hinter .ENDIF geschriebenen Anweisungen ausgeführt.

Die folgende Tabelle zeigt die Relation zwischen Parametern Op1 und Op2 und ihre Bedeutung:

Relation	Bedeutung
<Op1>,EQ,<Op2>	Op1 ist gleich Op2
<Op1>,NE,<Op2>	Op1 ist gleich Op2
<Op1>,GT,<Op2>	Op1 ist größer als Op2 in der ASCII-Sortierreihenfolge
<Op1>,LT,<Op2>	Op1 ist kleiner als Op2 in der ASCII-Sortierreihenfolge
<Op1>,GE,<Op2>	Op1 ist größer oder gleich Op2
<Op1>,LE,<Op2>	Op1 ist kleiner oder gleich Op2

Die Parameter Op1 und Op2 können Ketten, Variable oder konkatenierte Ketten sein. Ein numerischer Vergleich wird durchgeführt, wenn Op1 und Op2 beide numerisch sind. Andernfalls werden diese Ketten verglichen.

Zwischen .If und .ENDIF können beliebige andere SCI Primitiv Kommandos oder Aufrufe für andere Prozeduren stehen, einschließlich anderer .IF Anweisungen, d.h. geschachtelte bedingte Anweisungen dürfen eine Tiefe von 32 Schachtelungen nicht übersteigen.

Das folgende Beispiel zeigt die Anwendung der Primitiv Kommandos .IF, .ELSE und .ENDIF. In diesem Beispiel stellen die im CC Kommando (Copy Concatenate) geschriebenen Abkürzungen IAN und OAN die Pfadnamen der Eingabe- und Ausgabedateien dar. Die in der CC Prozedur benutzten Promptnamen sind INPUT ACCESS NAME(S) und OUTPUT ACCESS NAME.

```
.PROC EXP(EXAMPLE PROC)= 0,
          INPUT PATHNAME=ACNM,
          OUTPUT PATHNAME=ACNM,
          DELETE?=YESNO(N)
.IF &DELETE?,NE,"N"
 .IF &DELETE?,NE,"Y"
   MSG TEXT = "RESPONSE TO DELETE? MUST BE Y OR N"
   .EXIT
 .ENDIF
.ENDIF
CC IAN= &INPUT PATHNAME,
   OAN= &OUTPUT PATHNAME
.IF &DELETE?,EQ,"Y"
 DF PATHNAME(S)= &INPUT PATHNAME, ARE YOU SURE?=YES
.ENDIF
.EOP
```

In diesem Beispiel wird das Primitiv Kommando .IF dazu benutzt, die Benutzerantwort auf den DELETE? Prompt mit Y oder N zu vergleichen. Im Falle einer von Y oder N abweichenden Antwort wird die Meldung angezeigt und die Prozedur gibt die Kontrolle mittels Primitiv Kommando .EXIT an den SCI zurück.

Nachdem festgestellt wurde, daß der Wert von DELETE? entweder Y oder N ist, kopiert die Prozedur die Eingabedatei in die Ausgabedatei. Wurde als Antwort Y eingegeben, wird nach Ablauf der Copy-Operation die Eingabedatei mittels Kommandoprozedur FD (Delete File) gelöscht.

Dieses Beispiel zeigt die Verwendung eines SCI Kommandos in einer Prozedur. Das CC Kommando bewirkt das Kopieren einer Plattendatei in der gleichen Weise, als wäre das Kommando als Antwort auf einen SCI Prompt eingegeben worden. Kommandoprozedur CC hat 5 Prompts: INPUT ACCESS NAME? OUTPUT ACCESS NAME, REPLACE?, MAXIMUM RECORD LENGTH und FOREGROUND? (Y/N). IAN und OAN sind Abkürzungen für die ersten zwei Prompts und wurden unter Berücksichtigung der bereits beschriebenen Regeln für Abkürzungen gebildet. Zu beachten ist auch das Komma hinter IAN= &INPUT PATHNAME. Dieses Komma kennzeichnet die nächste Zeile als Forsetzungszeile des CC Kommandos. Die CC Prozedur setzt den Anfangswert des REPLACE? Prompts auf NO. Da der Kommandoaufruf dem REPLACE? Parameter keinen Wert zuweist, wird NO zur Vorbesetzung für REPLACE?. RE=Y könnte dazu verwendet werden, REPLACE? den Wert YES zuzuweisen. Der Prompt MAXIMUM RECORD LENGTH wird nicht benötigt, da er in der Prozedur als *INT definiert ist. Der Prompt FOREGROUND? (Y/N) wird auf den Anfangswert YES in der CC Prozedur gesetzt. Da der Kommando Aufruf FOREGROUND? (Y/N) nicht lädt, wird YES zum Standardwert.

Ruft eine andere Prozedur eine SCI Kommandoprozedur auf, so muß die aufrufende Prozedur alle nichtoptionale Prompts zur Verfügung stellen. Die Nichterfüllung dieser Bedingung hat die Ausgabe der Fehlermeldung 9011-REQUIRED ARGUMENT NOT PRESENT zufolge. Werden keine Prompts zur Verfügung gestellt, so werden die Standardprompts am Bildschirm angezeigt. Prozeduren, die im Hintergrund-Modus ablaufen (z.B. Batch) müssen alle notwendigen Prompts selbst versorgen.

Wenn eine Prozedur eine andere Prozedur aufruft, braucht der SCI zusätzlichen Speicherplatz für die in der neuen Prozedur enthaltenen Prompt. Der Fehler FF0B-KEYWORD TABLE OVERFLOW wird gemeldet, wenn die Schachtelungstiefe der Prozeduraufrufe zu hoch ist. Die zulässige Schachtelungstiefe hängt von der Anzahl und Größe der vorhandenen Prompts ab, die in den aufgerufenen Prozeduren vorhanden sind.

4.2.7 Primitiv Kommandos .LOOP, .UNTIL, .WHILE und .REPEAT

Primitiv Kommando .LOOP, .UNTIL, .WHILE und .REPEAT bewirken die iterative Wiederholung von SCI Anweisungsblöcken. Eine Schleife wird durch eine .LOOP Anweisung begonnen und mit einer .REPEAT Anweisung beendet. Primitiv Kommandos .UNTIL und .WHILE können beliebig oft und an beliebigen Stellen in dem zu wiederholenden Block vorkommen.

Die Primitiv Kommandos für iterative Schleifen haben folgendes Format:

```
.LOOPY
.UNTIL      <Op1>,<Beziehung>,<Op2>
.WHILE      <Op1>,<Beziehung>,<Op2>
.REPEAT
```

Parameter Op1 und Op2 können Ketten, Variable und konkatenierte Ketten sein. Der Relation Parameter gibt die Art der durchzuführenden Vergleichsoperation (numerischer Vergleich oder Vergleich von Zeichenketten) an. Die Grundstruktur einer Schleife in einer SCI Prozedur ist wie folgt:

.LOOP

⋮ SCI Anweisungen

.UNTIL or .WHILE

⋮ SCI Anweisungen

.REPEAT

⋮

Eine Schleife wird durch eine .LOOP Anweisung begonnen und mit einer .REPEAT Anweisung beendet. Die Schleife muß mindestens eine .WHILE oder .UNTIL Anweisung enthalten, welche an beliebiger Stelle stehen können und mehrfach auftreten dürfen. Die SCI Anweisungen innerhalb einer Schleife werden fortlaufend ausgeführt, bis die Bedingung in einer .WHILE Anweisung den Wert 'FALSCH' oder die Bedingung in einer .UNTIL Anweisung den Wert 'WAHR' erhält. Anschließend führt der SCI die erste Anweisung aus, die dem .REPEAT folgt.

Das folgende Beispiel zeigt die Verwendung der Primitiv Kommandos für iterative Schleifen.

```
.PROC EXP(EXAMPLE PROC)=0,
          INPUT PATHNAME=ACNM(@ $EX$IP),
          LISTING DEVICE=NAME(@ $PF$D),
          NUMBER OF COPIES?=INT(1)
          .SYN  $EX$IP= &INPUT PATHNAME
          .SYN  $PF$D= &LISTING DEVICE
          .SYN  NUM= &NUMBER OF COPIES?
.LOOP
  .UNTIL      @ NUM,LE,0
  PF          FILE=@ $EX$IP,L=@ $PF$D
  .EVAL       NUM = @ NUM-1
.REPEAT
          .SYN NUM=""
.EOP
```

In diesem Beispiel werden die Primitiv Kommandos .LOOP, .UNTIL, .EVAL und .REPEAT als Zählmechanismus verwendet.

Dem Synonym NUM wird die, als Antwort auf den Prompt NUMBER OF COPIES? eingegebene Ganzzahl zugewiesen. Die .UNTIL Anweisung vergleicht den Wert von NUM mit Null. Ist NUM größer als Null, wird Kommandoprozedur PF (Print File) angestoßen; diese Kommandoprozedur druckt den Inhalt der als Antwort auf INPUT PATHNAME eingegebenen Datei auf das als Antwort auf LISTING DEVICE eingegebene Gerät aus. Anschließend vermindert das Primitiv Kommando .EVAL den Wert von NUM um 1, und .REPEAT bewirkt die Wiederholung der Schleife. Die Schleife wird solange wiederholt, bis der Wert von NUM gleich oder kleiner Null ist. Dann werden die, dem Primitiv Kommando .REPEAT folgenden Primitiv Kommandos ausgeführt.

Das am Ende der Prozedur liegende Primitiv Kommando .SYN wird dazu benutzt, das Synonym NUM aus der Synonymtabelle zu löschen. Nach ihrer Beendigung müssen Prozeduren ihre internen Synonyme löschen, um den Überlauf der Synonymtabelle zu verhindern.

Der Benutzer darf als Antwort auf INT einen negativen Wert eingeben. Aus diesem Grund wird durch .UNTIL der Wert auf gleich oder kleiner Null verglichen.

Durch Verwendung der .LOOP Anweisung kann auch eine unbeabsichtigte Endlosschleife entstehen. Die fehlerhafte Benutzung von .WHILE oder .UNTIL kann zu Folge haben, daß eine neue Prozedur in einer Endlosschleife kreist, aus der es mittels Steuertasten am Terminal keinen Ausgang findet. Hat das System nur ein Terminal, kann durch Neuladen des Systems Abhilfe geschaffen werden. Handelt es sich um ein Multi-Terminal System kann ein Neuladen unerwünschte Konsequenzen haben. In diesem Fall kann ein zweites Terminal das STS Kommando (Show Task Status) aufrufen und unter Angabe der RUN ID und der Stationsnummer dieses Terminals, kann das KT Kommando (Kill Task) eingegeben werden, um den ablaufenden Task abzubrechen.

4.2.8 Primitiv Kommando .MENU

Das Primitiv Kommando .MENU bewirkt, daß der SCI bei seiner Rückkehr in den Kommando Modus das spezifizierte Menü am Bildschirm anzeigt. Ein Menüzyklus wird unmittelbar vor dem Anzeigen des SCI Prompts durchlaufen. Das Primitiv Kommando .MENU beeinflußt nur den nächsten Menüzyklus. Nachfolgende Menüzyklen kommen auf das Hautpmenü zurück. Dieses Primitiv Kommando hat folgendes Format:

.MENU [<Menüname>]

Es gibt drei Varianten des Parameters 'Menüname':

1. Es wurde kein Menü spezifiziert - Der Gebrauch des Primitiv Kommandos .MENU ohne Spezifizierung eines Menünamens bewirkt, daß der SCI den Menüzyklus vor dem Anzeigen des nächsten SCI Prompts umgeht.

2. Menüname - Wurde ein Menüname spezifiziert (1-6 alphanumerische Zeichen), so zeigt der SCI das angegebene Menü im nächsten Menüzyklus am Bildschirm an. Die Anzeige ist unabhängig davon, ob das Terminal sich im TTY oder VDT Modus befindet. Der SCI stellt dem Menünamen die Zeichen M$ voran, um den Namen der, in der Benutzerprozedur-Bibliothek(en) gespeicherten Datei, die das Menü enthält, zu erhalten.

3. Menüname - Wurde dem Menünamen ein * vorangestellt, so wird das Menü nur dann am Bildschirm angezeigt, wenn das Terminal im VDT Modus arbeitet.

Die Eingabe eines Schrägstrichs (/) als Antwort auf einen SCI Prompt wird vom SCI in .MENU konvertiert. Das Standardmenü,das weiter hinter beschrieben wird, zeigt die Verwendung dieses Zeichens. Der Terminal-Benutzer gibt /PDEV ein, um sich das Programmentwicklungs-Menü anzeigen zu lassen. Der SCI konvertiert den Schrägstrich in .MENU und die Eingabe wird dadurch zu .MENU PDEV.

/	ist gleichbedeutend mit .MENU
/ DEV	ist gleichbedeutend mit .MENU DEV
/*DEV	ist gleichbedeutend mit .MENU*DEV

Das folgende Beispiel zeigt die Verwendung des Primitiv Kommando .MENU:

```
.PROC   NM(NEW MENU PROC)
        .MENU   MYMENU
.EOP
```

Das im .MENU spezifizierte Menü wird nur einmal angezeigt. Um das durch den SCI Prompt angezeigte Vorgabemenü ändern zu können, muß der MENU Parameter des Primitiv Kommandos .OPTION herangezogen werden. Das Standard-Systemmenü hat den Namen .S$PROC.M$LC.

4.2.9 Primitiv Kommando .OPTION

Dieses Primitiv Kommando ermöglicht es dem Benutzer, einige grundlegende Schnittstelleneigenschaften des SCI zu ändern, um sie dadurch den örtlichen Sprach- und Anwendungsbedürfnissen anzupassen. Das Primitiv Kommando .OPTION hat folgendes Format:

.OPTION [PROMPT =<Zeichenkette>] [,MENU = <Name>] [,PRIMITIVES=<YES/NO>]

Die Parameterdefinitionen sind wie folgt:

Schlüsselwort	Zugeordneter Wert	Funktion
PROMPT	Eine alternative Prompt-Zeichenkette. Muß aus weniger als 50 Zeichen bestehen.	Ermöglich das Spezifizieren des SCI Prompts. Die Vorbesetzung für den SCI Prompt ist ( ) und wird durch ASCII-Codes dargestellt.
MENU	Name des Hauptmenüs. (Der SCI stellt dem angegebenen Menü automatisch die Zeichen M voran, um das Menü in Benutzerbibliothek PROC zu erhalten).	Ermöglicht die Bestimmung der Zulässigkeit oder Unzulässigkeit von Primitiv Kommandos als Antwort auf den SCI Prompt ( ). Primitiv Kommandos sind in Prozeduren auf jeden Fall erlaubt. Darüberhinaus unterbindet .OPTION PRIMITIVES = NO in einem Batch den weiteren Gebrauch von Primitiv Kommandos im gleichen Batch; Primitiv Kommandos sind jedoch trotzdem noch in Prozeduren erlaubt.
PRIMITIVES	YES oder NO	

Das folgende Beispiel zeigt die Verwendung des Primitiv Kommandos .OPTION für die Anzeige der Zeichen (KC) statt dem Standardprompt ( ).

```
.PROC EXP(NEW PROMPT PROCEDURE)
        .OPTION PROMPT = "(KC)"
.EOP
```

Abhängig vom Benutzer ID können Alternativ-Prompts auch am Bildschirm angezeigt werden durch Schreiben der folgenden .OPTION Anweisung in Datei M$00 der Prozedurbibliothek:

.OPTION PROMPT=ST@ $$ST USER:@ $$UI( )

Während der Benutzeranmeldung gibt dieses Kommando statt ( ) den folgenden Text aus:

ST.01 USER:JF0012 ( )

Die im letzten Beispiel geschriebenen Synonyme werden nur zum Zeitpunkt der Ausführung des Primitiv Kommandos .OPTION ausgewertet. Die zu diesem Zeitpunkt den Synonymen zugewiesene Werte werden bis zum Auftreten der nächsten .OPTION Anweisung bzw. bis zur Abmeldung und anschließender Anmeldung am Terminal als permanente Werte für den SCI Prompt betrachtet.

Die Prozedur .S$PROC.M$00 wird zur Zeit der Anmeldung am Terminal dazu benutzt, die Standardumgebung für das Terminal festzulegen. Das folgende Beispiel zeigt die .OPTION Anweisung für die Anmeldeprozedur:

.OPTION PROMPT = " SCI V 200 ( ) ",PRIMITIVES = NO

Dieses Kommando ändert den SCI Prompt vom "( )" in "SCI V 200 ( )" und macht Primitiv Kommandos der Primärstufe unwirksam.

Die Anmeldungsprozedur .S$PROC.M$00 kann modifiziert werden und so den individuellen Anwenderbedürfnissen angepasst werden.

4.2.10 Primitiv Kommando .OVLY

Das Primitiv Kommando .OVLY wird in TAXO Standardprozeduren verwendet. Es wird Benutzern empfohlen, .OVLY nicht zu benutzen, da die im Kommando enthaltenen Referenzen möglicherweise mit zukünftigen Erweiterungen des Betriebssystems inkompatibel sein könnten. Falsch kodierte .OVLY Aufrufe erhöhen auch die Wahrscheinlichkeit des Systemausfalls erheblich. Aus diesen Gründen wird die Verwendung von .OVLY durch den Benutzer nicht empfohlen.

VORSICHT

Der unzulässige Gebrauch der .OVLY Anweisung kann zu einem Systemzusammenbruch führen.

4.2.11 Primitiv Kommando .PROMPT

Dieses Kommando wird dazu benutzt, zusätzliche Information vom Benutzer anzufordern. .PROMPT kann zum Aufnehmen zusätzlicher Prompts benutzt werden, wenn eine größere Anzahl, als durch die .PROC Prompt-Liste angezeigt werden kann, benötigt wird. Mittels der .PROC Prompt-Liste kann am 911er Bildschirm bis 22 und am 913er Bildschirm bis 10 Prompts angezeigt werden. Das Primitiv Kommando .PROMP kann auch dazu verwendet werden, wahlfreie Benutzerantworten auf die normalen Prozedur-Prompts aufzunehmen. Die Syntax für dieses Primitiv Kommando ist wie folgt:

.PROMPT [(voller Name>)] [=Ganzzahl>] [,Prompt-Liste>]

Die durch .PROMPT definierten Prompts werden nicht am gleichen Schirm angezeigt, wie die durch .PROC definierten (d.h. der Bildschirm wird zuerst gelöscht und die neuen Prompts anschließend angezeigt).

Der Parameter 'voller Name' ist wahlfrei und definiert die Zeichenkette, die am Bildschirm angezeigt werden soll, wenn .PROMPT im interaktiven Modus ausgeführt wird. Der Parameter 'Ganzzahl' spezifiziert die niedrigste Privilegstufe, welche die Prozedur aufrufen darf. Diese Privilegstufe darf jedoch höher sein, als die in .PROC angegebene Privilegstufe. Der Parameter 'Prompts' ist eine Liste der Prompts, für die der Benutzer Antworten eingeben muß.

Das folgende Beispiel zeigt die Verwendung von .PROMPT:

```
.PROC EXP(EXAMPLE PROC)=0,
          INPUT PATHNAME=ACNM,
          OUTPUT PATHNAME=ACNM,
          DISPLAY OR COPY?=STRING(DISPLAY)
          .IF &DISPLAY OR COPY?,EQ,"DISPLAY"
           SF FILE= &INPUT PATHNAME
           .ELSE
           CC  IAN= &INPUT PATHNAME,
                OAN= &OUTPUT PATHNAME
.PROMPT (SUPPLEMENTARY QUESTION)=3
          DELETE FILE?=YESNO(N)
          .IF &DELETE FILE?,EQ,"Y"
           DF PATHNAME= &INPUT PATHNAME, ARE YOU SURE = YES
          .ENDIF
.EOP
```

In diesem Beispiel wird der Prompt DELETE FILE?, der zu den Zusatzabfragen gehört, nur dann am Bildschirm angezeigt, wenn der Benutzer im Begriff ist, die Datei zu kopieren und wenn die Prozedur durch einen Benutzer, dessen Privilegstufe mindestens 3 ist, aufgerufen wird. Ein Benutzer, der eine Privilegstufe hat, die niedriger ist als 3, erhält eine Fehlermeldung, wenn .PROMPT in der Prozedur aufgefunden wird und die Prozedur zu diesem Zeitpunkt abgebrochen wird.

4.2.12 Primitiv Kommando .SHOW

Dieses Primitiv Kommando zeigt den Inhalt der angegebenen Datei, bzw. die Inhalte der angegebenen Dateien, am Bildschirm eines interaktiven Terminals an. .SHOW hat folgendes Format:

```
.SHOW acnm[ ,acnm. . .]
```

'acnm' ist der Name einer Datei, oder ein Synonym, das auf eine Datei verweist.

.SHOW kann nicht dazu benutzt werden, eine Programm- oder Imagedatei am Bildschirm anzuzeigen. .SHOW arbeitet in der gleichen Weise wie das SF Kommando (Show File).

Das folgende Beispiel zeigt die Anwendung des Primitiv Kommandos .SHOW:

```
.PROC EXP(EXAMPLE SHOW FILE)=0,
         INPUT FILENAME=ACNM(@ $$F$P)
         .SYN $SF$P= &INPUT FILENAME
.SHOW @@$SF$P
.EOP
```

In diesem Beispiel wird .SHOW dazu verwendet, den Inhalt der in INPUT FILENAME angegebenen Datei am Bildschirm anzuzeigen. Der Anfangswert von INPUT FILENAME wird auf den Dateinamen der letzten Eingabedatei gesetzt. Durch das Primitiv Kommando .SYN wird der Name der letzten Eingabedatei durch den in dieser Prozedur als INPUT FILENAME angegebenen Dateinamen ersetzt.

Synonyme, die als Dateinamen-Prompts für .SHOW benutzt werden, müssen, wie im Beispiel gezeigt, mit zwei At-Zeichen (@) kodiert sein. Wird ein Promptname benutzt und kann ein Synonymname im Prompt eingegeben werden, so sollte dem Promptnamen das At-Zeichen (@) vorangestellt werden. Z.B. @ &INPUT FILENAME.

4.2.13 Primitiv Kommando .SPLIT

.SLPIT dient dazu, den ersten Ausdruck aus einer Werteliste zu entfernen. Dieses Primitiv Kommando hat folgendes Format:

.SPLIT LIST = <Zeichenkette>, FIRST = <Synonymname>, REST= <Synonymname>

Die rechts von LIST= angegebene <Zeichenkette> muß mit einer Klammer auf beginnen und mit einer Klammer zu enden. Die Wirkungsweise dieses Primitiv Kommandos wird durch folgendes Beispiel erläutert:

Vor der Ausführung des .SPLIT Kommandos		Nach der Ausführung des .SPLIT Kommandos	
Syntax von LIST	Wert des Synonyms als Syntaxbeispiel	Wert des Synonyms rechts von FIRST=	Wert des Synonyms rechts von REST=
LISTE=(A,B,C)	-	A	(B,C)
LIST=(A)	-	A	Null
LIST=( )	-	Null	Null
LIST=(X,Y),Z,G)	-	(X,Y)	(Z,G)
LIST=(@ SYN)	A,B,C	A	(B,C)
LIST=@ WXB	(A,B,C)	A	(B,C)
LIST=(@ XYZ)	(A,B,C)	(A,B,C)	Null
LIST=(@ B1)	A(,B,C)	Fehler	
LIST=(@ B2)	A,B,C	Fehler	

Die Werte in der Werteliste müssen voneinander durch Komma getrennt sein. Runde Klammern können benutzt werden, um die Aufteilung der Liste zu bestimmen.

Das folgende Beispiel zeigt die Verwendung von .SPLI in einer Kommandodefinition:

```
.PROC EXP(EXAMPLE PROC),
        INPUT PATHNAME=ACNM(@ $EX$IP),
        OUTPUT PATHNAME(S)=(ACNM)
        .SYN $EX$IP= &INPUT PATHNAME
        .SYN $EX$OP= (&OUTPUT PATHNAME)
        .LOOP
            .SPLIT LIST=@ $EX$OP,
            FIRST=$EX$P,
            REST=$EX$OP
            CC IAN=@ $EX$IP, OAN=@ $EX$P
        .WHILE @ $EX$OP,NE,$EX$OP
        .REPEAT
.EOP
```

Im obigen Beispiel wird die als Antwort auf den Prompt INPUT PATHNAME angegebene Datei in die, als Antwort auf den Prompt OUTPUT PATHNAMES(S) angegebene(n) Datei(en) kopiert. .SPLIT wird innerhalb einer Schleife verwendet, um auf die Ausgabedatei zugreifen zu können, zu welcher die Eingabedatei kopiert werden soll. Nach Erstellung einer Kopie schaltet .SPLIT auf die nächste Ausgabedatei um.

Das Primitiv Kommando .WHILE wird innerhalb der Schleife dazu benutzt, um festzustellen, ob die Eingabedatei in alle angegebene Ausgabedateien einkopiert worden ist. Dies ist der Fall, wenn das Synonym EXOP einen Nullwert erreicht hat.

Die runden Klammern, die im .SYN die Angabe &OUTPUT PATHNAME umschliessen, sind für einen Listenwert benötigt.

Das Primitiv Kommando .WHILE sollte wie im Beispiel gezeigt kodiert werden, d.h., dem ersten Synonymnamen sollte ein At-Zeichen (@) vorangestellt werden und dem zweiten Synonymnamen sollte kein At-Zeichen (@) vorangestellt werden. Wenn der erste Operand, d.h. @ EXOP, wegen Leerwerdens der Liste sich in einen Nullwert auflöst, so wird dieser Wert zum Synonymnamen, d.h., EXOP. Der zweite Operand in der .WHILE Anweisung wird als Literal behandelt, da ihm kein Vorzeichen vorangestellt wurde. Zu diesem Zeitpunkt würden die zwei Operanden identisch, d.h. gleich sein und die Schleife wird verlassen.

4.2.14 Primitiv Kommando .STOP

Das Primitiv Kommando .STOP meldet den Benutzer beim SCI ab, wenn es in einer im Vordergrund ausführenden Prozedur vorkommt. Wird es in einem Batch geschrieben, so beendet es den Batch. .STOP hat folgendes Format:

.STOP [TEXT=<Zeichenkette>] [,CODE=<Ganzzahl>]

Die durch den TEXT Parameter spezifizierte Zeichenkette, sowie der Wert für CODE, sind beide wahlfrei. Der TEXT Parameter kann dazu benutzt werden eine statt der Meldung BATCH SCI HAS COMPLETED anzuzeigende Zeichenkette an den Vordergrung SCI zu überbeben. Der TEXT = Parameter darf eine Prozedurzeile nicht überschreiten. Der CODE Parameter kann dazu verwendet werden, das in der Synonymtabelle des Vordergrund SCI gespeicherte Synonym $$BC nach Beendigung des Batch auf einen 4-stelligen hexadezimalen Wert zu setzen. Das Synonym $$BC des ausführenden SCI wird auf Null gesetzt, wenn eine Hintergrund-Task aktiviert wird. Ein im Hintergrund ausführender Task kann mittels des CODE Parameters dem Synonym $$BC in der Vordergrund-Synonymtabelle einen Wert zuweisen. Der Wert wird Synonym $$BC erst dann zugewiesen, wenn eine im Vordergrund ausführende Prozedur beendet wird. Aus diesem Grund kann das Synonym $$BC von der Prozedur, die ursprünglich den Hintergrund-Task aufgerufen hat, nicht benutzt werden. Die Parameter TEXT und CODE werden ignoriert, wenn der SCI sich nicht im Batch-Modus befindet.

Texte, die nicht fertig editiert wurden, gehen verloren, wenn .STOP im Vordergrund-Modus ausgeführt wird.

Das folgende Beispiel zeigt den interaktiven Gebrauch von .STOP:

```
,PROC EXP(TERMINAL IS LOGGING OFF)=0                    !FULL NAME DISPLAYED
        .STOP                                                   !TERMINATE SCI
.EOP
```

In diesem Beispiel wurde .STOP in einer Kommandoprozedur benutzt, um den SCI zu beenden und das Terminal abzumelden. Die Arbeitsweise von .STOP ist in diesem Fall identisch mit der Arbeitsweise des Q-Kommandos (Quit SCI).

Nach Beendigung eines Batch übergibt der SCI die im TEXT Parameter angegebene Meldung an den Vordergrund SCI des Terminals, welches das Batch vorgelegt hat. Führt das Terminal gerade ein WAIT Kommando aus, so zeigt der SCI die TEXT-Meldung sofort am Bildschirm an. Steht der SCI Prompt am Bildschirm, so kann die Meldung durch Drücken der RETURN-Taste zur Anzeige gebracht werden. Führt das Terminal ein Kommando im Vordergrund-Modus aus, so wird die Meldung nach Beendigung der Vordergrund-Task am Bildschirm angezeigt.

4.2.15 Primitiv Kommando .SCV

Das .SVC Primitiv Kommando wird in TAXO Standardprozeduren verwendet. Außerdem wird jedoch empfohlen, .SVC nicht zu benutzen, da die im Kommando anfallenden Referenzen mit künftigen Erweiterungen unkompatibel sein könnten.
Falsch kodierte .SVC Aufrufe erhöhen die Wahrscheinlichkeit eines Systemausfalls erheblich. Aus diesen Gründen wird die Verwendung durch den Benutzer nicht empfohlen.

4.2.16 Primitv Kommando .SYN

.SYN weist den Synonymen der Synonymtabelle den Wert zu und hat folgendes Format:

.SYN<Name>="<Wert>". . .

Der 'Name'-Parameter spezifiziert den Namen eines Symbols ohne das At-Zeichen (@). Der 'Wert'-Parameter kann eine Zeichenkette, eine Variable oder ein konkatenierter Ausdruck sein. Wertelisten müssen in runden Klammern eingeschlossen sein.

Wird einem Synonym ein Nullwert ("") zugewiesen, wird das Synonym aus der Synonymtabelle gelöscht. Synonyme, die nicht mehr benötigt werden, sollten grundsätzlich am Ende einer Prozedur gelöscht werden, damit die Synonymtabelle nicht unnötig gefüllt wird.

Das folgende Beispiel zeigt die Verwendung von .SYN in einer Kommandodefinition:

```
.PROC EXP(EXAMPLE PROC)=0,
          INPUT PATHNAME(S)=(ACNM)@ $EX$IP),
          OUTPUT PATHNAME=ACNM(@ $EX$OP),
.SYN $EX$IP= (&INPUT PATHNAME)
.SYN $EX$OP= &OUTPUT PATHNAME
          CC IAN=@ $EX$IP,
             OAN=@ $EX$OP
.EOP
```

Dieses Beispiel weist den Synonymen EXIP und EXOP Werte zu. Andere Kommandoprozeduren dürfen auch auf diese Synonyme zugreifen. Das Synonym EXIP ist der Name der letzten Eingabedatei und EXOP der Name der letzten Ausgabedatei. Prozeduren dürfen diesen Synonymen Werte zuweisen, auf die die nachfolgenden Tasks zugreifen können.

4.2.17 Primitiv Kommando .USE

.USE spezifiziert die, vom SCI zu benutzenden Prozedurbibliotheken. Das Primitiv Kommando .USE hat folgendes Format:

.USE<acnm1>[, <acnm2>]

Nach Ausführung einer .USE Anweisung werden sämtliche Kommandprozeduren und Menüs von den durch< acnm1> und <acnm2 >angegebenen Bibliotheken genommen. Die durch < acnm1> angegebene Bibliothek wird zuerst durchsucht. Kann die Prozedur bzw. das Menü in dieser Bibliothek nicht gefunden werden, wird die durch< acnm2> spezifizierte Bibliothek durchsucht. Das .USE Kommando bleibt solange wirksam, bis ein neues .USE Kommando ausgeführt wird, oder bis sich das Terminal abmeldet und sich anschließend wieder anmeldet. Die Ausführung eines .USE Kommandos ohne Operanden übergibt die Kontrolle erneut an die Standard System Bibliothek. In diesem Fall ist die Vorbesetzung für< acnm1> .S$PROC und <acnm2> ist Null.

Wird eine Kommandoprozedur interaktiv mittels .PROC installiert, speichert der SCI die Kommandodefinition in das durch< acnm1> spezifizierte Directory.

Eine Warnung wird nach Aufrufen des .USE Kommandos ausgegeben, wenn nicht eine der Bibliotheken das durch .OPTION spezifizierte Hauptmenü beinhaltet. Dieses Menü ist normalerweise die Datei .S$PROC.M$LC. Wird ein SCI Kommando als Antwort auf den Prompt eingegeben, bildet der SCI den Dateinamen der Kommandobibliothek durch Verketten von< acnm1>, und dann von< acnm2> mit dem vom Benutzer eingegebenen Prozedurnamen. Zum Beispiel:

```
( ) .USE .S$PROC, .USERLIB
( ) EX
```

bewirkt, daß der SCI eine Kommandodatei namens .S$PROC.EX sucht.

Wird diese nicht gefunden, sucht der SCI die Datei .USERLIB.EX. Das .USE Primitive Kommando richtet Directories der Prozedurbibliothek ein.

MERKE

Kann nach Aufrufen des .USE Kommandos das Standardmenü nicht gefunden werden, wird eine Warnung am Bildschirm angezeigt. Ein Menü wird nicht angezeigt. Kommt dies in einer Prozedur vor, so wird diese sofort abgebrochen. Eine der im .USE Kommando angegebenen Prozedurbibliotheken sollte eine Datei enthalten, in der das Hautpmenü gespeichert ist.

Die Ausführung eines .USE Kommandos ohne Parameter gibt die Kontrolle an die System Prozedur Bibliothek zurück (.S$PROC).

Die Datei .S$PROC.M$00 wird oft verwendet, um Prozedurbibliotheken mittels des .USE Kommandos aufzurufen. Zum Beispiel:

```
.PROC M$00
  .USE .S$PROC, .USERLIB
  .IF @$$UI,EQ,JF0012
    .USE .S$PROC, .JFLIB
    .OPTION MENU=JF
  .ENDIF
.EOP
```

Diese Prozedur, die unter .S$PROC.M$00 gespeichert ist, würde während der Anmeldeprozedur eines beliebigen Benutzers aufgerufen werden. Die Prozedur stellt die Prozedurbibliothek .USERLIB allen Benutzern zur Verfügung, mit Ausnahme des Benutzers, dessen ID JF0012 ist. Benutzer JF0012 würde die Prozedurbibliothek .JFLIB benutzen und auch mit dem Standardmenünamen von M$JF arbeiten.
Die Prozedur M$00 kann auf diese Weise verwendet werden, um den SCI benutzerspezifisch zu gestalten.

5 AUSFÜHRUNG DES SCI

Die Techniken, die der SCI für die Bildung von Datei- und Directoynamen, für die Ausführung von Batches und für die Herstellung von Prozessorschnittstellen benutzt werden, sind in den nachfolgenden Abschnitten beschrieben.

5.1 Namensbildung für Dateien und Directories

Zwei Faktoren müssen für die Bildung von Prozedurnamen berücksichtigt werden. Erstens dürfen diese nicht mit den in der Standard SCI Prozedurbibliothek bereits vorhandenen Prozedurnamen in Konflikt kommen. Zweitens, eine Prozedur kann in die System Prozedur Bibliothek .S$PROC abgespeichert oder eine neue Prozedurbibliothek für die vom Benutzer beschriebenen Prozeduren erstellt werden.

Ein Konflikt zwischen bereits existierenden Prozedurnamen und geplante Erweiterungen können durch Ausführung des LD Kommandos (List Directory) zwecks Durchsicht von .S$PROC wie folgt vorgebeugt werden:

1. Das LD Kommando als Antwort auf den SCI Prompt eingeben.

2. Den Pfadnamen .S$PROC aus Antwort auf den ersten Prompt eingeben.

3. Eine Druckereinheit, z.B. LP01, bzw. den Namen einer Plattendatei, z.B. .LIST, als Antwort auf den zweiten Prompts (LISTING ACCESS NAME) eingeben. RETURN drücken ohne vorherige Eingabe falls kein Ausdruck erwünscht ist. Obwohl ein Ausdruck nicht unbedingt nötig ist, dient er doch als Hilfsmittel, welches den Benutzer über die in der System Prozedur Bibliothek vorgenommenen Änderungen informiert.

4. Wurde kein Ausdruck angefordert, wird das Directory am Bildschirm des Terminals angezeigt. A (eine Zeile nach oben) und B (eine Zeile nach unten) können zur Durchsicht des ganzen Directory verwendet werden.

5. Sollte z.B. eine Prozedur namens AX der Bibliothek hinzugefügt werden, so kann der Benutzer durch die Bildschirmanzeige feststellen, ob diese bereits existiert. Existiert diese Prozedur noch nicht, kann die neue Prozedur unter dem Namen AX in die Bibliothek aufgenommen werden.

MERKE

Prozedurnamen können bis zu 8 Zeichen lang sein.

Es ist nicht notwendig, eine separate Bibliothek für die, vom Benutzer geschriebenen Prozeduren anzulegen. Die Bibliothek .S$PROC kann mit nur gerinfügigen Schwierigkeiten benutzt werden. Wird ein Directory-Ausdruck von .S$PROC angefordert, so werden sämtliche Directory-Einträge aufgelistet. Sollte dies unerwünscht sein, kann das Problem durch Anlegen einer separaten Bibliothek für Benutzerprozeduren gelöst werden. Eine separate Bibliothek würde den Aufruf des .USE Kommandos für den Bibliotheknamen benötigen. Der Gebrauch von separaten Prozedurbibliotheken läßt auch den Zugriff zu diesen Prozeduren auf bestimmt Benutzer beschränken.

Eine Bibliothek für Benutzerprozeduren wird wie folgt angelegt:

1. CFDIR (Create File Directory) als Antwort auf den SCI Prompt eingeben.

2. Den Namen der Prozedurbibliothek als Pfadnamen eingeben. Z.B., .USERLIB kann als Namen der Bibliothek für Benutzerprozeduren verwendet werden. Dem Namen muß ein Punkt vorangestellt werden, um die Bibliothek auf die Systemplatte zu speichern. Dem Punkt kann der Plattenanme einer Ersatzplatte vorangestellt werden. .USERLIB soll nur ein Beispiel eines möglichen Bibliotheknamens darstellen. Der Benutzer darf der Bibliothek jeden Namen, der mit bereits vorhandenen Directory-Namen nicht in Konflikt kommt, für dieses Zweck verwenden.

3. Die maximale Anzahl Prozeduren, die voraussichtliche in diese Bibliothek übernommen werden sollen, als Antwort auf den Prompt MAX ENTRIES eingeben. Diese sollte die erwartete Anzahl um mindestens 10% übersteigen, um die Zugriffszeit auf ein Minimum herabzusetzen.

4. RETURN drücken um den dritten Prompt (Satzlänge) zu umgehen.

Die in der Benutzerbibliothek zu speichernden Prozeduren können jetzt, unter Angabe von .USERLIB.NAME als Pfadname der Ausgabedatei, mittels des Text Editors erstellt werden. NAME ist der Prozedurname.

Um den SCI aufzufordern, den Namen .USERLIB zu verwenden, muß ein .USE Kommando zum Aufrufen des Bibliotheknamens ausgeführt werden. Dies geschieht in der Form:

.USE .S$PROC, .USERLIB

Das .USE Kommando kann entweder interaktiv am Terminal eingegeben oder in einer Prozedur z.B. in Anmeldeprozedur .S$PROC.M$00, aufgerufen werden.

5.2 Stapelverarbeitung (Batch Streams)

Ein Batch Stream wird zur Steuerung von Hintergrund Tasks benutzt. Das Format eines Batch ähnelt dem Format einer Kommandoprozedur mit Ausnahme der Tatsache, daß die benötigten Prompts im Batch kodiert werden müssen, weil Hintergrund-Prozeduren auf das Terminal nicht zugreifen können. Normalerweise ist die erste Zeile eines Batchstroms ein BATCH Kommando und die letzte Zeile ein EBATCH Kommando.

Ein Batch wird auf eine Platte in der gleichen Weise gespeichert, wie eine Kommandoprozedur. Das in einem fürheren Abschnitt aufgeführte Beispiel für das Editieren einer Prozedur veranschaulicht dieses Verfahren.

Ein Batch wird durch ein XB Kommando (Execute Batch) aufgerufen. Das XB Kommando kann entweder als Antwort auf den SCI Prompt oder als Teil einer Kommandoprozedur eingegeben werden. In beiden Fällen muß aber das XB Kommando mit zwei Prompts versorgt werden: dem Input Access Name und dem Listing Acccess Name. Der Input Access Name ist der Name der Batchdatei. Der Listing Access Name ist entweder das Kennzeichen für eine Druckereinheit oder der Pfadname der Plattendatei, auf welche die Batch-Listeing ausgegeben werden kann.

5.2.1 Beispiel für eine Batch-Kommandofolge

Die im Folgenden gezeigte Batch Kommandofolge wurde erstellt, um die von einem Lohnabrechnungsprogramm in Plattendatum erstellten Lohnlisten mit Hilfe des Spoolers auszudrucken.

```
BATCH                    !Batch zum Lohnlisten Druck
PF FILE =.LOHNLST1, LD = LP01
PF FILE =.LOHNLST2, ANSI=Y, LD=LP01
PF FILE =(.LOHNLST3, .LOHNLST4), LD=LP01
PF FILE =.LOHNLST5, LD=LP02
EBATCH TEXT ="Batch zum Lohnlisten Ausdruck beendet"
```

Diese Batch Kommandofolge kann durch eine Kommandoprozedur mit dem Namen LOHNDRU aufgerufen werden. Diese Prozedur hat folgendes Aussehen:

```
.PROC
LOHNDRU (Drucken der Lohnlisten)
XB,
IAN = .LOHN.LOHNDRU
LAN = .LDRBTCHL
MSG TEXT="Ausdruck gestartet"
.EOP
```

5.2.4 Die Batch-Listing

```
 <0001> BATCH
USER ID                          GGE022
STATION ID                       ST02
BATCH INPUT ACCESS NAME          .PAYRLIB.PRTPAYB
BATCH LISTING ACCESS NAME        .PRBTCH
LS (LIST SYNONYMS) ?             NO
16:11:05 TUESDAY, MAR 29, 1983.
 <0002> SPDEV
 <0003> PF FILE=.PAYROL1, LD=LP01
FILE PATHNAME(S)                 .PAYROL1
ANSI FORMAT?                     NO
LISTING DEVICE                   LP01
DELETE AFTER PRINTING?           NO
NUMBER OF LINES/PAGE             ** NULL **
FORM                             ** NULL **
PREFERRED PRINTING?              NO
NUMBER OF COPIES                 1
SHOW OUTPUT STATUS               YES
 FORM LIST

  DEVICE   FORM
 END OF FORM LIST


 PRINT QUEUE

   ST/  USER  DEVICE   FORM DEL  SPOOL FILE NAME
   PRI   ID                        ID
   A/1 GGE022  LP01            N  S00027 .PAYROL1
 END OF PRINT QUEUE
 <0004> PF FILE=.PAYROL2, ANSI=Y, LD=LP01
FILE PATHNAME(S)                 .PAYROL2
ANSI FORMAT?                     Y
LISTING DEVICE                   LP01
DELETE AFTER PRINTING?           NO
NUMBER OF LINES/PAGE             ** NULL **
FORM                             ** NULL **
PREFERRED PRINTING?              NO
NUMBER OF COPIES                 1
SHOW OUTPUT STATUS               YES
 FORM LIST

  DEVICE   FORM
 END OF FORM LIST


 PRINT QUEUE

   ST/  USER  DEVICE   FORM DEL  SPOOL FILE NAME
   PRI   ID                        ID
   A/1 GGE022  LP01            N  S00028 .PAYROL2
 END OF PRINT QUEUE
 <0005> PF FILE=(.PAYROL3, .PAYROL4), LD=LP01
FILE PATHNAME(S)                 .PAYROL3, .PAYROL4
ANSI FORMAT?                     NO
LISTING DEVICE                   LP01
DELETE AFTER PRINTING?           NO
NUMBER OF LINES/PAGE             ** NULL **
FORM                             ** NULL **
PREFERRED PRINTING?              NO
```

```
NUMBER OF COPIES                    1
SHOW OUTPUT STATUS                  YES
 FORM LIST

  DEVICE    FORM
 END OF FORM LIST


 PRINT QUEUE

   ST/   USER   DEVICE    FORM DEL   SPOOL FILE NAME
   PRI    ID                           ID
   A/1 GGE022   LP01            N   S00028 .PAYROL2
   W/1 GGE022   LP01            N   S00029 .PAYROL3
   W/1 GGE022   LP01            N   S00030 .PAYROL4
 END OF PRINT QUEUE
 <0006> PF FILE=.PAYROL5, LD=LP02
FILE PATHNAME(S)                    .PAYROL5
ANSI FORMAT?                        NO
LISTING DEVICE                      LP02
DELETE AFTER PRINTING?              NO
NUMBER OF LINES/PAGE                ** NULL **
FORM                                ** NULL **
PREFERRED PRINTING?                 NO
NUMBER OF COPIES                    1
SHOW OUTPUT STATUS                  YES
 FORM LIST

  DEVICE    FORM
 END OF FORM LIST


 PRINT QUEUE

   ST/   USER   DEVICE    FORM DEL   SPOOL FILE NAME
   PRI    ID                           ID
   A/1 GGE022   LP02            N   S00031 .PAYROL5
 END OF PRINT QUEUE
 <0007> EBATCH TEXT="THE PAYROLL REPORT BATCH IS FINISHED"
LS (LIST SYNONYMS) ?                NO
TEXT                                THE PAYROLL.REPORT BATCH IS FINISHED
CODE                                ** NULL **
```

5.2.3 Die Ausführung des Batch

Nach Eingabe von LOHNDRU als Antwort auf den SCI Prompt müssen folgende Schritte ausgeführt werden:

1. Der SCI zeigt den vollen Prozedurnamen "DRUCKEN DER LOHNLISTEN" am Bildschirm an, führt ein paar Verarbeitungsschritte durch und zeigt anschließend den Text "AUSDRUCK GESTARTET" im unteren Teil des Bidlschirms an. Der Benutzer muß RETURN drücken, um zum SCI zurückzugelangen.

2. LOHNDRU hat die XB Prozedur aufgerufen und ihr die zwei Prompts IAN und LAN übergeben.

3. Der SCI wendet die Regeln für die Abkürzung von Feldnamen an, um IAN dem Prompt INPUT ACCESS NAME und LAN dem Prompt LISTING ACCESS NAME zuzuweisen. Die Tatsache, daß dem Promptttyp (ACNM) kein Stern vorangestellt wurde und die Tatsache, daß ihnen kein Anfangswert zugewiesen wurde bezeichnet diese Prompts als zwingend.

4. Die XB Prozedur weist die Werte von IAN und LAN den Synonymen XBI und XBL zu. Es ist zu bemerken, daß die BATCH Prozeduren diese Synonyme benutzen, um den Pfadnamen des Batch und das Listengerät zu identifizieren.

5. XB ruft das Primitiv Kommando .QBID auf, um die Task zu aktivieren. .QBID startet die Task im Hintergrund-Modus. Diese Anweisung startet die eigentliche Ausführung des Hintergrund-Tasks.

6. XB (der immer noch im Vordergrund-Modus ausgeführt wird) löscht die zwei Synonyme die für das Übergeben von Werten an die Hintergrund-Task verwendet wurden. Der Hintergrund-Task wurde zur Zeit seiner Aktivierung eine Kipie der Synonymtabelle zur Verfügung gestellt. Die Synonyme sind im Vordergründ gelöscht, stehen jedoch der Hintergrund-Task immer noch zur Verfügung. Das Löschen der Synonyme verhindert einen Überlauf der Synonymtabelle.

7. Nach Löschen der Synonyme wird XB beendet und der SCI Prompt wird am Bildschirm des Terminals erneut angezeigt.

8. Auf .LOHN.LOHNDRU wurde im Hintergrund als Batch zugegriffen.

9. Die erste Zeile des LOHNDRUB Batch ruft die BATCH Prozedur auf. Diese Prozedur beinhaltet keine zwingenden Prompts. Ihre Prompt-Liste zeigt, daß sie die Standard SCI Synonyme $$UI und $$ST verwendet. Diese stellen die User ID und die Stationsnummer zur Verfügung, welche dann in der Batch Listing erscheinen. XBI und XBL, die in XB generiert wurden, werden jetzt vom BATCH benutzt, um auf den Pfadnamen des Batch und auf das Listgerät zuzugreifen.

10. Der im Batch enthaltende Prompt LS kann auf "Y" gesetzt werden, wenn eine Auflistung der Synonymtabelle als Teil der Batch-Listing erwünscht wird. In LOHNDRUB würde BATCH LS = Y in der ersten Zeile diese Synonymliste anfordern.

11. Die BATCH Prozedur bestätigt seine Ausführung im Hintergrund-Modus durch Prüfen des Synonyms $$MO. BATCH beendet sich selbst, wenn es sich herausstellt, daß sie nicht im Hintergrund-Modus war.

5.3 Expert Mode

Die Kommandoeingabe im Expert Mode ist vergleichbar mit dem Kommandoaufruf in Batch-Kommandofolgen, d. h., ein Kommando kann interaktiv eingegeben werden, indem der Kommandoname, gefolgt von einem Zwischenraum und dann alle Keywords mit den ihnen zugeordneten Werten, eingetastet wird. Dabei kann das letzte Zeichen einer Zeile ein Fortsetzungszeichen sein(",," oder "="), was bedeutet, daß der Kommandoaufruf durch weitere Eingaben fortgesetzt wird.

ANMERKUNG:

Wortzuweisungen von Keywords sind nur für das gerufene Kommando möglich. Werden während der Kommandoabarbeitung Unterprozeduren aufgerufen, so werden Wortzuweisungen an Keywords der Unterprozeduren nicht automatisch übergeben.

Ist ein Kommandoaufruf im Expert Mode vollständig eingegeben, gibt der SCI am Bildschirm die Keyword Maske aus, wobei jedoch die erste Zeile, die normalerweise den vollen Kommandonamen enthält, unterdrückt wird. Würde bei der Kommanoeingabe alle Pflichteingaben spezifiziert und sind sie syntaktisch richtig, wird das Kommado sofort ausgeführt. Ein Beispiel soll die Kommadoeingabe im Expert Mode verdeutlichen:
Wurde für ein Keyword, bei dem eine Pflichteingabe besteht, kein Wert angegeben und existiert auch keine Vorbesetzung dafür, wird für dieses Keyword in der Keyword Maske eine Eingabe angefordert und erst dann die Kommandoausführung gestartet.

```
( ) SF FILE = .S$NEWS
```

Durch diese Eingabe wird die Datei .S$NEWS am Benutzerterminal angezeigt, ohne daß der SCI eine Eingabe für das Keyword File PATHNAME des SF Kommandos anfordert. Es sei darauf hingewiesen, daß Keywords wie in Batches eindeutig abgekürzt werden können (im Beispiel: FILE Pathname wird abgekürzt durch FILE). Würde nun das SF Kommando erneut aufgerufen, hätte das Keyword FILE PATHNAME die Vorbesetzung .S$NEWS. Um diese Datei jedoch erneut anzuzeigen, kann der Benutzer jetzt auch das SF Kommando, gefolgt von einem Punkt, eintasten:

```
( ) SF.
```

Die Datei .S$NEWS wird wieder angezeigt. Daraus folgt, wenn bei der Kommadoeingabe hinter dem Kommando ein Punkt eingegeben wird, nimmt der SCI an, daß der Benutzer ein Expert Mode arbeitet. Die Zuweisung FILE = .S$NEWS im ersten Beispiel und der Punkt "," im zweiten Beispiel veranlassen den SCI im Expert Mode zu arbeiten.

Der Expert Mode im SCI erlaubt also eine schnellere Kommandoeingabe, wenn alle Keywords bereits vorbesetzt sind oder mit Standarwerten für optimale Parameter gearbeitet werden soll. Eine weitere wichtige Funktion besteht darin, sog. "versteckte" Keywords mit Werten zu versorgen. Versteckte Keywords sind solche, die zwar für ein Kommando definiert sind, in der Keyword Maske jedoch nicht erscheinen, weil für sie kein Typ festgelegt ist.

Eine weitere Eigenschaft des Expert Modes besteht darin, daß Keywords, die beim Kommandoaufruf zwar angegeben wurden, denen aber kein Wert zugewiesen wurde, automatisch den Wert "Y" (für Yes) erhalten.

Beispiel:

Das SCI Kommando DCOPY besitzt ein verstecktes Keyword WITHOUT DEFAULTS. Hat dieses Keyword den Wert "Y", wird der Save Mode von DCOPY aktiviert, andernfalls der Standardkopiervorgang. Ein Aufruf des Save Modes von DCOPY kann nun auf eine der folgenden Arten erfolgen:

```
( )  DCOPY WITHOUT DEFAULTS = YES
( )  DCOPY W = YES
( )  DCOPY WITHOUT DEFAULTS
( )  DCOPY W
```

Diese Eigenschaft des Expert Modes sollte allerdings nur mit Vorsicht angewandt werden, und nur dann, wenn der Benutzer mit der Funktion des Kommandos vertraut ist.

Eine weitere Eigenschaft des Expert Modes ist es, "versteckten" Prompts beim Kommandoaufruf Werte zuzuweisen. Versteckte Prompts sind solche, für die in der Prozedurdefinition kein Typ festgelegt ist. Solche versteckten Prompts können spezielle Funktionen eines Kommandos veranlassen, die für den Benutzer normal nicht notwendig sind. Als Beispiel hierfür sei die DCOPY Prozdur genannt:

```
DCOPY (DISK COPY AND RESTORE),
    WITHOUT DEFAULTS
.IF & WITHOUT DEFAULTS, GE, "Y"
    Bid DCOPY Task for Save Mode
.ELSE
    Bid DCOPY Task for Copy Mode
.ENDIF
```

Wird dieses Kommando (wie allgemein üblich) mit

```
( ) DCOPY
```

aufgerufen, ist für das Prompting WITHOUT DEFAULT kein Wert vorhanden (d.h &WITHOUT DEFAULTS ist kleiner als "Y") und die Kommandos hinter .ELSE werden ausgeführt. Erfolgt der Aufruf jedoch mit

```
( ) DCOPY WITHOUT DEFAULTS = YES
```

so ist die .IF Bedingung erfüllt und die Kommandos vor .ELSE werden ausgeführt.